



FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

**ALGORITMO PARA DOTAR DE PROACTIVIDAD A UN SISTEMA DE
RESPUESTA AUTOMÁTICO A INTRUSIONES EN REDES INFORMÁTICAS
USANDO MODELOS OCULTOS DE MARKOV Y PROGRAMACIÓN
EVOLUTIVA**

PARA OPTAR EL TITULO PROFESIONAL DE INGENIERO DE SISTEMAS

AUTOR

José Emilio Satornicio Chambergo

ASESOR

Mg. Víctor Alexci Tuesta Monteza

PIMENTEL, JULIO DE 2018

**ALGORITMO PARA DOTAR DE PROACTIVIDAD A UN SISTEMA DE
RESPUESTA AUTOMÁTICO A INTRUSIONES EN REDES INFORMÁTICAS
USANDO MODELOS OCULTOS DE MARKOV Y PROGRAMACIÓN
EVOLUTIVA**

Aprobación de la tesis

Satornicio Chambergó José Emilio
Autor

Ing Heber Ivan Mejia Cabrera
Asesor Metodológico

Mg. Víctor Alexci Tuesta Monteza
Asesor Especialista

Ing. Heber Ivan Mejia Cabrera
Presidente de Jurado

Ing. Alex Franklin Coronado Navarro
Secretario de Jurado

Mg. Víctor Alexci Tuesta Monteza
Vocal de Jurado

DEDICACION

Dedico este trabajo al amor, apoyo, tolerancia y esfuerzo de todas las personas que han estado a mi lado a lo largo de estos años, empezando por mi adorada madre María Susana, mi hermana Roció del Milagro y mi amada esposa Luz Estela.

A mis amigos del grupo de complementación académica de la escuela de Ingeniería de Sistemas con quienes a lo largo de estos años hemos compartido más que un aula, el esfuerzo por ver triunfar nuestras metas.

Y sobre todo, a Dios y a la vida por haberme dado esta oportunidad de crecer como persona y profesionalmente en esta área del conocimiento.

¡Muchas Gracias!

AGRADECIMIENTO

Son muchas las personas que han estado a mi lado a lo largo de estos años, y que han hecho posible que pueda estar escribiendo estas líneas en este momento. Entre todos han conseguido que llegue al final del camino. Gracias

En primer lugar debo dar las gracias muy especialmente a mi asesor de Tesis, Mg. Víctor Alexci Tuesta Monteza. Su trabajo, apoyo y valiosa orientación han hecho posible la finalización de este trabajo.

También quisiera expresar mi gratitud al Ing. Iván Mejía Cabrera, quien con sus valiosas enseñanzas y exigencia en los cursos impartidos, hicieron que nos acostumbremos a hacer las cosas cada vez mejor y a no rendirnos.

Gracias también al Dr. Anastacio Antolino Hernandez de nacionalidad mexicana quien desinteresadamente brinda su apoyo al desarrollo de este proyecto. Al Dr. Gregorio Martínez Pérez y Msc. Manuel Gil Pérez ambos de nacionalidad española quienes brindaron su apoyo cuando les fue solicitado y con quienes actualmente compartimos un grupo de desarrollo colaborativo.

Gracias también a Nilton Guevara Palomino, un gran amigo por las amanecidas de estudio, charlas y sobremesas, que han hecho que los momentos duros no lo parecieran tanto

A mi familia, por su apoyo incondicional a lo largo de toda mi vida. Sé que a veces ha sido difícil y que en muchos momentos ni yo mismo me hubiera aguantado, pero ustedes siempre ahí, haciendo que todo pareciera mucho más fácil. Gracias por su paciencia y comprensión y por sacarme una sonrisa en todo momento.

Gracias Luz Estela por el apoyo, la comprensión y el cariño que me has demostrado durante el prolongado y en ocasiones difícil periodo de tiempo que he estado dedicado a la realización de esta Tesis, en especial en estos últimos meses. Gracias por ser esa persona que me aporta ese punto de alegría, optimismo y tranquilidad que a veces me falta, y por hacer que aprenda a ver la vida de otra forma.

Contenido

CAPÍTULO 1: PROBLEMA DE INVESTIGACION..... 18

1.1. Situación Problemática:..... 18

1.2. Formulación del Problema: 20

1.3. Delimitación de la Investigación 22

1.4. Justificación e Importancia de la Investigación: 22

1.5. Limitaciones de la Investigación:..... 23

1.6. Objetivos de la Investigación:..... 23

CAPÍTULO II: MARCO TEORICO..... 24

2.1 Antecedentes de Estudios..... 24

2.2 Estado del arte 28

2.3 Bases Teórico Científicas..... 33

2.3.1 Seguridad en Redes 34

2.3.2 Metodología de Intrusión 37

2.3.3 Tipos de Ataques Informáticos y Acciones de Respuesta

Asociadas..... 41

2.3.4 Sistemas de Respuesta a Intrusiones (IRS) 79

2.3.5 Modelos Ocultos de Markov (HMM) 88



2.3.6 Algoritmos genéticos y HMMs.....	111
2.3.7 Programación evolutiva.....	112
Evolución de HMMs a través de Algoritmos Evolutivos	114
Arquitectura de un HMM a evolucionar	115
Algoritmo “Evolution”	119
Algoritmo GenInitPop.....	124
Algoritmo InitChrome	125
Algoritmo Initialization	127
Algoritmo Induction.....	129
Algoritmo Forward.....	131
Algoritmo FuncProbMult.....	132
Algoritmo AddTransition.....	133
Algoritmo DeleteTransition.....	135
Algoritmo MutParameters	136
Algoritmo MutTransitions	139
Algoritmo DeleteState	140
Algoritmo AddState	142



2.4	Definición de terminología.....	144
CAPITULO III: MARCO METODOLÓGICO.....		146
3.1	Tipo y diseño de la investigación	146
3.2	Población y muestra.....	146
3.3	Hipótesis	146
3.4	Operacionalización.....	147
3.5	Métodos, técnicas e instrumentos de recolección de datos	148
3.6	Procedimiento para la recolección de datos.....	149
3.7	Plan de análisis estadístico de datos	149
3.8	Criterios éticos	151
3.9	Criterios de rigor científico.....	151
CAPÍTULO IV: ANALISIS E INTERPRETACION DE RESULTADOS.....		152
4.1	Resultados en tablas y gráficos	152
CAPÍTULO V: PROPUESTA DE INVESTIGACION		157
5.1	Método	158
5.1.1	Comparación de AIRS existentes	160
5.1.2	Seleccionar el AIRS adecuado	162



5.1.3 Analizar y entender la arquitectura del AIRS seleccionado 164

5.1.4 AIRS sin Proactividad 181

5.1.5 Analizar y Preparar Batería de Ataques 181

5.1.6 Inicio del Diseño y Construcción de Modulo de Proactividad 190

5.1.7 Analizar y entender HMM y PE 190

5.1.8 Determinar y Aplicar Algoritmos para evolucionar HMM 194

Obtención de HMM optimizados 201

5.1.9 201Obtención de HMM optimizados

5.1.10 Propuesta de arquitectura de AIRS Proactivo 202

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES 251

6.1 Conclusiones 251

6.2 Recomendaciones 253

Referencias 254



Índice De Figuras

Figura 1. Ciclo de Seguridad del Proceso de Seguridad.....	37
Figura 2. Fases de la Metodología de la Intrusión.....	38
Figura 3. Relación entre respuestas proactivas y reactivas sobre el marco temporal de un ataque.	83
Figura 4. Taxonomía de Sistemas de Respuesta a Intrusiones.	87
<i>Figura 5. Representación de un HMM con 4 Estados.....</i>	<i>99</i>
Figura 6. Etapa de Inducción del Algoritmo de Forward.....	104
Figura 7. Recursión del Algoritmo de Viterbi.....	106
Figura 8. El paso Backtracing del Algoritmo de Viterbi.....	107
Figura 9. Definición de Parámetros en el Proceso Evolutivo.....	122
Figura 10. HMMCM evolucionado con el ancho de banda de entrada y salida	154
Figura 11. Consumo real anómalo del ancho de banda de entrada y salida..	155
Figura 10. Método aplicado para el desarrollo de la propuesta de Investigación.	159
Figura 11. Arquitectura de Proyecto RECLAMO.....	165



Figura 12. Representación grafica de la extensión “IntrusionTrust” 167

Figura 13. Evolución de un HMM con EP 201

Figura 25. Algoritmo de Predicción 240

Índice de Tablas

Tabla 2. Gen que representa la Matriz de Transiciones.....	116
Tabla 3. Gen que guarda la Matriz de Medias del Cromosoma.	117
Tabla 4. Gen que mantiene la Matriz de Covarianzas para cada Estado.....	118
Tabla 5. Gen con el Vector Pi del Cromosoma	118
<i>Tabla 7. Criterios de rigor científico.....</i>	<i>151</i>
Tabla 8. Resultado de la detección usando HMM basado en Baum-Welch...	152
Tabla 9. Detección de anomalías usando HMMCU basado en EP	153
Tabla 12. Arquitectura de un HMM o Cromosoma.....	196
Tabla 13. Gen que representa la Matriz de Transiciones.....	196
Tabla 14. Gen que guarda la Matriz de Medias del Cromosoma.	197
Tabla 15. Gen que mantiene la Matriz de Covarianzas para cada Estado.....	197
Tabla 16. Gen con el Vector Pi del Cromosoma	198
Tabla 17. Requisitos de entrada del AIRS Proactivo.....	205
Tabla 18. Requisitos de salida del AIRS Proactivo	205
Tabla 22. Tráfico de datos de salida de la red de la municipalidad distrital de Chongoyape (Kbit/segundo).....	284



Resumen

La seguridad de la información dentro de una organización se ha convertido en una necesidad importante y cada vez más creciente. Debido al continuo incremento en la complejidad y sofisticación de los ataques cibernéticos y actividades maliciosas, el aumento de su velocidad de difusión y la lentitud de reacción frente a las intrusiones existentes en la actualidad, surge la necesidad de implementar mecanismos de detección y respuesta a intrusiones, que detecten y además sean capaces de bloquear el ataque, y mitiguen su impacto en la medida de lo posible.

Muchos componentes de seguridad se pueden instalar como parte de la arquitectura de una red informática (IDS, IPS, IRS, firewall, etc.) de tal manera que se pueda monitorear todos los eventos en la red. Uno de los componentes de seguridad de mayor investigación en los últimos años son los Sistemas de Respuesta Automática frente a Intrusiones (AIRS), cuya característica deseable es la proactividad la cual consiste en la capacidad del sistema de reaccionar frente a una intrusión antes de que esta comprometa al recurso objetivo llegando incluso a la capacidad de prever la ocurrencia de la intrusión y actuar en consecuencia intentando controlarlo y prevenirlo mediante la ejecución de una acción de respuesta, sin embargo esta característica aun es deseable en los IRS existentes.

Este trabajo de tesis hace frente al problema existente y propone una arquitectura proactiva para un AIRS, basado en el consumo de ancho de banda de red utilizado en la Municipalidad Distrital de Chongoyape. Se determina en primera instancia el modelo de comportamiento normal del consumo de ancho de banda, de acuerdo a los datos obtenidos en el centro de cómputo de dicha entidad. El modelo generado se utiliza en la detección de anomalías o variaciones en el consumo de ancho de banda de la red. La presencia de una anomalía indicara si se está haciendo mal uso de la red (presencia de virus, gusanos, denegación de servicio, etc.), para su diseño se propone el uso de

Modelos Ocultos de Markov (HMM), los cuales serán optimizados de manera automática utilizando la programación evolutiva, las variables a utilizar principalmente serán la de consumo de ancho de banda de red de entrada y la de consumo de ancho de red de salida, pero cabe destacar que el sistema está diseñado para analizar dos o más métricas del comportamiento del sistema. Los resultados preliminares del experimento indican que la arquitectura propuesta es eficiente para el monitoreo de anomalías o intrusiones en tiempo real y que identifica perfectamente anomalías que puedan poner en peligro la red.

Palabras clave: Proactividad, Intrusión, Predicción, Sistema de Respuesta Automática a Intrusiones, Modelos Ocultos de Markov

Abstract

The information security within an organization has become an important and increasingly growing need. Due to the continuous increase in complexity and sophistication of cyber attacks and malicious activities, increasing its rate of diffusion and slow reaction to existing intrusion today, the need to implement mechanisms to detect and respond to intrusions arises, to detect and also be able to block the attack, and mitigate its impact on the extent possible.

Many safety components can be installed as part of the architecture of a computer network (IDS, IPS, IRS, firewall, etc.) so that it can monitor all network events. One of the security components further research in recent years are the systems Automatic Response to Intrusions (AIRS), whose desirable feature is the proactivity which is the system's ability to react to an intrusion before it is commit to target resource even to the ability to predict the occurrence of intrusion and act accordingly trying to control it and prevent it by executing a response action, however this feature is even desirable in existing IRS.

This thesis faces the existing problem and proposes a proactive architecture for a AIRS, based on consumption of network bandwidth used in the Municipalidad Distrital de Chongoyape. Normal behavior pattern of bandwidth consumption, according to data from the computer center of the entity is determined in the first instance. The generated model is used in detecting anomalies or variations in the consumption of bandwidth of the network. The presence of an anomaly indicate whether misusing network (presence of viruses, worms, denial of service, etc.) to design using Hidden Markov Models (HMM) is proposed, which will be optimized automatically using evolutionary programming, variables to be used mainly be the consumption of bandwidth network input and consumption of network bandwidth output but it is noteworthy that the system is designed to analyze two or more metrics the system behavior. Preliminary results of the experiment indicate that the proposed architecture is efficient for monitoring anomalies



or intrusions in real time and perfectly identifies anomalies that may endanger the network..

Keys Words: Proactivity, Intrusion, Prediction, Automated Response Intrusion System, Hidden Markov Model

Introducción

Los Sistema de detección de intrusiones (IDS) monitorean los eventos de red para detectar actividades maliciosas o cualquier intento de entrar o comprometer un sistema. Se suele proporcionar alertas de mala calidad, que son insuficientes para apoyar la rápida identificación de anomalías en curso o predecir el próximo objetivo o paso de anomalía. Además, las alertas de mala calidad innecesariamente hacen que el sistema sea declarado vulnerado, posiblemente desencadenando respuestas de prevención de alto impacto.

Sistema de Respuesta de intrusiones (IRS), es el siguiente nivel de tecnología de seguridad (Jemili, F., Zaghdoud, M., & Ben Ahmed, M, 2007). Su misión es ejecutar buenas estrategias para prevenir el crecimiento de anomalías y devolver un sistema al modo saludable. Proporciona la seguridad en todos los niveles del sistema, como los paquetes de datos del kernel del sistema operativo y de red. Aunque se han propuesto muchos IRSs, diseñando buenas estrategias para una respuesta eficaz de las anomalías ha sido siempre una preocupación. Es necesario un equilibrio entre la degradación del rendimiento del sistema y la máxima seguridad. De acuerdo con el nivel o grado de automatización, los sistemas de respuesta de intrusión se pueden clasificar como: sistemas de notificación, los sistemas de respuesta manuales y sistemas de respuesta automatizada (Stakhanova, Basu, & Wong, 2007). Los Sistemas de respuesta Automatizada (AIRS) tratan de ser totalmente automatizado utilizando los procesos de toma de decisiones sin intervención humana. El principal problema de este enfoque es la posibilidad de ejecutar una respuesta inadecuada en caso de un problema. Los AIRS se pueden dividir en: 1) el modelo estático: mapea



una alerta a una respuesta predefinida. Este modelo es fácil de construir, pero la principal debilidad es que las medidas de respuesta son predecibles. 2) modelos dinámicos: las respuestas se basan en varios factores, como el estado del sistema, las métricas de ataque (Frecuencia, gravedad, confianza, etc.) y la política de la red.

En otras palabras, la respuesta a un ataque no puede ser la misma función, por ejemplo en el host de destino. Una desventaja de este modelo es que no aprende nada de los ataques a los ataques, por lo que el nivel de inteligencia sigue siendo el mismo hasta la próxima actualización. 3) Costo sensible: es una técnica interesante que trata de armonizar el daño de intrusiones y el coste de respuesta. Para medir el daño intrusión, se necesita un componente de evaluación de riesgo.

En este contexto, las contribuciones de este trabajo incluyen: (1) la definición de un módulo para la proactividad en presencia de sofisticados ataques de varios pasos y evitar que mediante la ejecución de conjuntos de apropiadas respuestas usando Modelos Ocultos de Markov y Programación Evolutiva para reducir el tiempo de entrenamiento y el uso de memoria, este módulo se puede aplicar en una red real para reaccionar de manera proactiva ante cualquier tipo de ataques.

Este trabajo está organizado de la siguiente manera: en primer lugar, vamos a discutir el trabajo relacionado y varios métodos existentes para la proactividad. El modelo propuesto se ilustra en el Capítulo V. En capítulo III, se presentan los resultados experimentales. Conclusiones y trabajo futuro serán discutidos en capítulo VI.



CAPÍTULO 1: PROBLEMA DE INVESTIGACION

1.1. Situación Problemática:

La investigación en seguridad de la información, y en particular en sistemas para detectar y responder de forma automática a intrusos, se han convertido en un tema de actualidad y pertinencia en cuanto las redes son cada día más susceptibles a los ataques, esto debido a la sensibilidad de la información que viaja a través de ellas, según el Reporte de Amenazas de Seguridad de Internet de abril del 2014 emitido por (Symantec, 2014), el total de fuga de datos en el mundo en el año 2013 aumento un 62% respecto al año 2012, el mismo informe sitúa a nuestro país en el octavo lugar con mayor actividad maliciosa.

Una encuesta hecha por (Kaspersky, 2014), demuestra que el número total de ataques dirigidos ha ido en aumento. El 12% de las empresas que reportó un ataque dirigido en el año 2014 ha aumentado 9% del promedio reportado en 2013 y 2012.

El (Institute for Security and Open Methodolog) en su Manual de la Metodología Abierta de Testeo de Seguridad (OSSTMM por sus siglas en inglés) define como factor crítico a probar la “Seguridad en las tecnologías de Internet”, la cual incluye los sondeos de red y la identificación de sistemas como puntos clave, motivo adicional que muestra la importancia de la seguridad en las redes y la ventaja que puede ofrecer para prevenir ataques.

El campo de la seguridad en redes ha requerido de una constante investigación para evaluar y optimizar los mecanismos de control de



acceso y protección de la información, que permitan mitigar los ataques. Tradicionalmente, se emplean como parte de la arquitectura de una red informática firewalls y routers para la aplicación de políticas de seguridad y control de acceso respectivamente; sin embargo, estos medios de defensa perimetral no garantizan aun una protección del 100% contra los ataques existentes.

Por otro lado debido a la complejidad y sofisticación de los ataques informáticos así como el aumento de su velocidad de difusión y la lenta respuesta de reacción frente ellos, surge la necesidad de contar con mecanismos de detección y respuesta a intrusiones, que predigan, detecten y además sean capaces de bloquear un ataque y minimicen su impacto en la medida posible.

(VILLAGRÁ GÓNZALES, 2009) Una intrusión ocurre cuando se explota una determinada vulnerabilidad de una red de datos. Ante la imposibilidad de prever todas las vulnerabilidades que una red presenta, se hace necesario detectar los usos maliciosos que de estas se hacen, siendo esta la misión de los Sistemas de Detección de Intrusos (IDS), así mismo además de detectar un ataque de forma rápida y eficaz, es importante saber cómo reaccionar apropiadamente ante un determinado ataque, en que momento, quienes son los actores responsables y que acciones podrían llevarse a cabo.

1.2. Formulación del Problema:

En cualquiera de los casos citados anteriormente, las consecuencias de un ataque se traduce en un atentando contra la integridad, confidencialidad y disponibilidad de la información, ya sea que esta esté almacenada y sean accesibles desde Internet o que transite a través de ella.

En éste sentido, el campo de la seguridad en redes ha requerido de una constante investigación para evaluar y optimizar los mecanismos de control de acceso y protección de la información en tránsito, que permitan mitigar los ataques.

(Stakhanova, Basu, & Wong, 2007) Propone un IRS automático, sensible a costes, adaptativo y preventivo. Su modelo propuesto se basa en detectar comportamientos anómalos en los sistemas mediante la monitorización de su comportamiento en términos de llamadas del sistema, y en responder automáticamente cuando corresponda. Para ello, utiliza un mapeo entre recursos del sistema, acciones de respuesta y patrones de intrusión o anomalía (proponen el uso de un grafo de comportamiento normal y anómalo), que debe ser definido previamente por el administrador.

(Robayo Santana, 2009), en su tesis “Detección De Intrusos En Redes De Telecomunicaciones IP Usando Modelos Ocultos De Markov”, propone un esquema basado en dos características, la capacidad de clasificar tráfico de protocolos conocidos y la capacidad de detectar intrusiones en tráfico, mediante la aplicación de un Modelo Oculto de Markov, su investigación



demonstró tener una capacidad de detección de ataques e intrusiones superior al 95%, lo que lo hace una muy buena alternativa como producto de investigación.

(Antolino Hernandez, 2011), en su tesis doctoral “Detección de Anomalías en Redes de Computo utilizando Modelos Ocultos de Markov y Programación Evolutiva”, propone un esquema de detección de anomalías, basado en el consumo de ancho de banda de red mediante el cual determina un modelo de comportamiento normal de uso de ancho de banda, el cual una vez generado se utiliza en la detección de anomalías o variaciones en el consumo de ancho de banda de red. Para ello usa la técnica de Modelos Ocultos de Markov los cuales se crean y evolucionan a través de la Programación Evolutiva.

Como se ha evidenciado, hay investigaciones que están tratando de solucionar el problema planteado como las expuestas anteriormente, sin embargo aún hay ciertas limitaciones respecto al tema de la característica de Proactividad que debe tener un Sistema de Respuesta Automática a intrusiones para poder reaccionar frente a una intrusión antes de que esta comprometa al recurso objetivo, y es en este contexto donde se plantea la finalidad de la presente investigación, que propone la implementación de un algoritmo para dotar de Proactividad a un AIRS usando la técnica de Modelos Oculto de Markov y la Programación Evolutiva para su creación, evolución y entrenamiento.



1.3. Delimitación de la Investigación

El presente trabajo incluye el análisis de secuencias de observación que contienen el consumo de ancho de banda de entrada y el de salida de una semana completa del centro de datos de la Municipalidad Distrital de Chongoyape, la herramienta para realizar el monitoreo en el centro de datos de la municipalidad es PRTG Traffic Grapher V6.2.2.984, así mismo comprende la implementación de un algoritmo basado en Modelos Ocultos de Markov los cuales serán entrenados y evolucionados con Programación Evolutiva, el software utilizado en el desarrollo de los algoritmos y las pruebas del experimento será Mathematica versión 8.0.

1.4. Justificación e Importancia de la Investigación:

La seguridad en las redes informáticas es un tema muy relevante hoy en día y que despierta gran importancia para buscar mecanismos que garanticen que la información que viaja a través de ella no sea vulnerada por terceros con distintas intenciones, es por ello que la presente investigación contribuirá a conseguir un cierto grado de seguridad a los datos que viajan a través de una red, por otro lado en el tema de ingeniería se está aportando nuevo conocimiento con la aplicación y pruebas científicas de técnicas de control estocástico y programación evolutiva que servirán para seguir avanzando en esta línea de investigación.

1.5. Limitaciones de la Investigación:

La dificultad para obtener el código fuente de un Sistema de Respuesta Automática a Intrusiones no está disponible para tomarla y modificarla para los fines de la presente investigación.

Así mismo la falta de especialistas en la presente línea de investigación en la región Lambayeque y el país ha dificultado el asesoramiento en el desarrollo de la presente investigación, sin embargo se ha logrado conseguir asesoramiento de especialistas extranjeros.

1.6. Objetivos de la Investigación:

Objetivo general

Implementar un Algoritmo para dotar de proactividad a un sistema de respuesta automática a intrusiones en redes informáticas usando modelos ocultos de Markov y programación evolutiva.

Objetivos específicos

- A. Analizar los tipos de Ataques a una red.
- B. Determinar Algoritmos para la Mitigación del Ataque
- C. Aplicar los Algoritmos.
- D. Evaluar resultados.

CAPÍTULO II: MARCO TEORICO

2.1 Antecedentes de Estudios

En esta sección se describen los trabajos previos que integren el área de Seguridad Informática, específicamente de los Sistemas de Respuesta Automática a Intrusiones, así como las diferentes técnicas de inteligencia artificial que se pueden aplicar para dar solución a la problemática descrita.

(Robayo Santana, 2009), **Detección de Intrusos en Redes de Telecomunicaciones IP Usando modelos ocultos de Markov**

Este trabajo plantea un modelo con dos características, la capacidad de clasificar tráfico de protocolos conocidos y la capacidad de detectar intrusiones en tráfico. Esto alineado con la necesidad de generar esquemas novedosos para garantizar niveles altos de seguridad de la información, los cuales, en cooperación con otros esquemas, brinden protección diferentes para los usuarios de redes de telecomunicaciones, en especial, Internet. En ese orden de ideas esta propuesta de modelo de detección es un esquema sustentado en la aplicabilidad de los **Modelos Ocultos de Markov** la cual demostró ser una muy buena alternativa para la detección de intrusos y el análisis de tráfico desconocido en redes que funcionen sobre la pila de protocolos IP. El esquema de entrenamiento propio de los Modelos Ocultos de Markov hace que los periodos de tiempo necesarios para la generación de modelos nuevos y la re-calibración de modelos antiguos (medidos desde la cantidad de iteraciones necesarias para el entrenamiento) sean muy breves permitiendo la constante evolución del modelo al momento de la inserción de nuevas características en los protocolos, como lo pueden ser



versiones nuevas o implementaciones disímiles. El esquema propuesto de utilización de las frecuencias de tamaños de las conversaciones en los diferentes protocolos para la generación de secuencias y el alineamiento con el Modelo, demostró tener una capacidad de detección de ataques e intrusiones superior al 95%, lo que lo hace una muy buena alternativa como producto de investigación.

Sin embargo este esquema aun no opera bajo un esquema de tiempo real de detección en el cual se definan mecanismos para realizar el agrupamiento de secuencias y el cálculo de histogramas, así como también la posibilidad de realizar un estudio que determine cantidad y características que optimicen el entrenamiento de los modelos, tanto en tiempo como capacidades.

(Antolino Hernandez, 2011), **Detección de Anomalías en Redes de Cómputo utilizando Modelos Ocultos de Markov y Programación Evolutiva**

Esta propuesta utiliza Modelos Ocultos de Markov, como modelos estocásticos probabilísticos y la Programación Evolutiva para evolucionarlos, y utilizar los modelos evolucionados en la detección de anomalías.

Este trabajo a diferencia de muchos otros que utilizan Modelos Ocultos de Markov y los optimizan usando Algoritmos Genéticos, para el reconocimiento de voz o secuencia de ADN, utiliza secuencias multivariadas de valores reales, y se aplica para la detección de anomalías presentes en cualquier secuencia de observación.

El sistema es capaz de modelar la dinámica de un sistema, utilizando Modelos Ocultos de Markov de una variable (Univariado) o de un conjunto



de ellas (Multivariado), y modelar el comportamiento probabilístico normal actual del sistema en cuestión. Para la actualización del modelo, sólo es necesario volver a ejecutar nuestro sistema con la nueva serie de observación, para crear y entrenar el modelo con las nuevas series de observación.

Se concluye que los Modelos Ocultos de Markov generados con Programación Evolutiva son capaces de distinguir entre comportamiento normal y comportamiento anormal. Según sus experimentos muestran que los Modelos Ocultos de Markov Univariados realizan bien su actividad, en un 75% en la detección de anomalías.

Sin embargo este sistema no cuenta con la optimización de tiempo de procesamiento, tampoco se da una completa evolución de los Modelos Ocultos de Markov Multivariados.

(Guaman Loachamin, 2013), **Propuesta de Arquitectura e Implementación de un Módulo de Ejecución de acciones de Respuesta para un Sistema Autónomo de respuesta a Intrusiones basado en Ontologías**

Este estudio presenta una arquitectura con un ejecutor de respuestas el cual involucra a 6 componentes: los IDSs, se trabajó con NIDS Snort y HIDS OSSEC quienes se encargan de la detección de las intrusiones; el razonador AIRS, infiere la respuesta óptima a través de la evaluación de varias métricas de respuesta; el MCER, el módulo central de ejecución se encarga de construir una solicitud de respuesta y ubicar a los agentes de ejecución; el módulo de comunicación, establece una conexión confiable y segura entre



el MCER y los agentes de ejecución; los agentes de ejecución, ejecutan funciones de autenticación, listas blancas de ejecución, y la ejecución misma de la acción de respuesta; el componente de seguridad, que representa el dispositivo que lleva a cabo la acción de respuesta real utilizando su interfaz de línea de comandos, tal que se altere su funcionamiento tan pronto como es ejecutado.

Sin embargo cuando se requiere ejecutar una respuesta compuesta, es el MCER quien debe ejecutar cada acción simple de forma independiente; no obstante, en determinadas circunstancias podría ser útil que un agente de ejecución tenga la capacidad de ejecutar una acción sobre otro agente de ejecución. Esta funcionalidad podría implementarse como un plugin adicional, para evitar modificar la arquitectura original.

(Mateos Lanchas, 2013), **Contribución a la Automatización de Sistemas de Respuesta Frente a Intrusiones Mediante Ontologías**

Propone la arquitectura de un Sistema de Respuesta Automática a Intrusiones basado en ontologías, usando lenguajes formales de especificación de comportamiento y mecanismos de razonamiento, con el fin de garantizar coherencia semántica en un entorno heterogéneo, de tal forma que el Sistema de Respuesta Automática a Intrusiones tenga la capacidad de entender la sintaxis y semántica de las alertas de intrusiones generadas por diferentes IDS.

Si bien esta arquitectura de AIRS cumple con la mayoría de los requisitos que según (Stakhanova, Basu, & Wong, 2007) debería tener un AIRS, aun no cumple con el requisito de Proactividad de tal manera que pueda ejecutar



una acción de respuesta frente a una intrusión antes de que esta se lleve a cabo.

La presente investigación seguirá esta línea de investigación y buscara dotar de comportamiento proactivo a la arquitectura planteada por (Mateos Lanchas, 2013), aplicando algoritmos basados en técnicas de Modelos Ocultos de Markov y Programación Evolutiva.

2.2 Estado del arte

En este capítulo se hará una descripción del estado del arte de los trabajos basados en los Modelos Ocultos de Markov (HMM) y Programación Evolutiva aplicados a la Proactividad.

La utilización de los Modelos Ocultos de Markov inicialmente estaba orientada a secuencias de ADN, reconocimiento de voz, etc. Sin embargo en la actualidad existen trabajos orientados a detección de anomalías en redes. Si bien no se ha evidenciado aún el uso de Modelos Ocultos de Markov en la Proactividad respecto a las intrusiones en redes informáticas, si existen trabajos de investigación orientados a la predicción en otros campos como la medicina, la economía, etc. así mismo existen trabajos de investigación donde la Programación Evolutiva ha sido usada como técnica para la predicción.

A continuación se describen los trabajos relacionados al área en investigación los cuales se presentaran en orden cronológico descendente.

(Antolino Hernandez, 2011), plantea un esquema de detección de anomalías basado en el consumo de ancho de banda de red. Determina un modelo de

comportamiento normal de ancho de banda, el cual será el patrón para ser utilizado en la detección de anomalías o variaciones en el consumo de ancho de banda de la red. La presencia de una anomalía indicaría que se está haciendo mal uso de la red (virus, denegación de servicios, etc). Las variables utilizadas son la del consumo de ancho de banda de red de entrada y la del consumo de ancho de banda de red de salida. La técnica utilizada por este trabajo son los Modelos Ocultos de Markov Univariados y Multivariados los cuales evolucionan a través de los siguientes algoritmos evolutivos: Algoritmo Evolution, Algoritmo GenInitPop, Algoritmo InitChrome, Algoritmo Initialization, Algoritmo Induction, Algoritmo Forward, Algoritmo FuncProbMult, Algoritmo AddTransition, Algoritmo DeleteTransition, Algoritmo MutParameters, Algoritmo MutTransitions, Algoritmo DeleteState, Algoritmo AddState, logrando optimizar los parámetros de los HMM, para luego utilizar estos HMM evolucionados en la detección de anomalías

Se concluye que los Modelos Ocultos de Markov generados con Programación Evolutiva son capaces de distinguir entre comportamiento normal y comportamiento anormal. Según sus experimentos muestran que los Modelos Ocultos de Markov Univariados realizan bien su actividad, en un 75% en la detección de anomalías.

Como aportes para el presente proyecto de investigación se tomara en cuenta la técnica empleada para evolucionar y entrenar los Modelos Ocultos de Markov a través de los algoritmos de la Programación Evolutiva.

(Mesa, 2010), estudia los Modelos Ocultos de Markov (HMM) como predictores de genes específicamente de genes VSG en Trypanosoma



brucei. La técnica utilizada para el entrenamiento del HMM es el algoritmo de Baum-Welch a través del cual se estiman los parámetros del HMM, una vez establecidos los parámetros utiliza el algoritmo de Viterbi el cual estima, para cada secuencia de testeo, el camino de estados más probable según las estimación de los parámetros obtenidos por Baum-Welch, determinando así cuales regiones de las secuencias de testeo son predichas como Genes VSG.

En su investigación concluye que los Modelos Ocultos de Markov utilizados, resulto ser eficiente en la búsqueda de genes VSG, superando en todas las pruebas practicadas el 90% prediciendo todos los genes objetivo.

Como aportes para el presente proyecto de investigación se tomara en cuenta el Algoritmo de Viterbi empleado en la predicción.

(Diego Evin, Alejandro Hadad, Mauro Martina, & Bartolomé Drozdowicz, 2010), estudia la utilización de modelos ocultos de Markov para predecir estados de hipotensión en pacientes internados en unidades de cuidados intensivos. El procedimiento de predicción desarrollado cuenta con dos modelos de Markov, uno entrenado con datos fisiológicos de pacientes que en un determinado intervalo de tiempo desarrollan estados de hipotensión, y otro entrenado con datos de pacientes en los cuales no se registra dicho cuadro. Utiliza HMM de cuatro estados en el que cada estado está conectado por una probabilidad de transición a cualquier otro estado. En este caso para encontrar una secuencia óptima de estados utiliza el Algoritmo de Viterbi y al igual que en otros trabajos de investigación para el ajuste de parámetros recurre al Algoritmo de Baum-Welch el cual emplea un procedimiento



iterativo comenzando con probabilidades iniciales preasignadas y las ajusta dinámicamente con base en las secuencias observadas en el conjunto de entrenamiento.

Se concluye entonces que una de las técnicas más usadas para el entrenamiento en predicción de Modelos Ocultos de Markov es el Algoritmo de Viterbi, así mismo emplea el algoritmo de Baum-Welch para la optimización de parámetros del Modelo Oculto de Markov

(Jabbour & Maldonado, 2009), plantea una arquitectura llamada ETOM (Expertos Temporales Ocultos de Markov) la cual consiste en un Modelo Híbrido basado en Redes Neuronales y Modelos Ocultos de Markov, una característica distintiva de los ETOM es que la transición de estados de la serie de tiempo se modela a través de un Modelo Oculto de Markov, en el cual la matriz de transición de estados es variante en el tiempo.

El objetivo de esta investigación consistió en evaluar el desempeño de los ETOM en la predicción de índices bursátiles, y realizar una comparación con resultados obtenidos mediante RNA puras.

En esta ocasión como técnica inicial para el entrenamiento del HMM se usa un Algoritmo de Baum-Welch modificado, para los fines de investigación, por otro lado para cada uno de los patrones obtenidos se usa una Red Neuronal Multicapa de conexión hacia adelante para desempeñar el papel de experto en el respectivo estado. Hecho esto se genera la Arquitectura ETOM la cual también es entrenada aplicando el Algoritmo Clustering Difuso de C-Medias (CDCM), utilizando como criterio de agrupación, la situación dinámica asociada a cada patrón.

Los experimentos fueron realizados con 15 índices bursátiles, y se obtuvo que los modelos ETOM superan sustancialmente a las RNA tanto en precisión como en su capacidad para capturar patrones.

Como aporte para este trabajo de investigación, se analizara al detalle esta Arquitectura que si bien es un tanto compleja ha dado buenos resultados para la predicción. Resaltando la modificación del Algoritmo de Baum-Welch el cual será analizado al detalle.

(Romero Morales, Ventura Soto, & De Castro, 2009), este estudio usa algoritmos evolutivos como técnica de minería de datos para el descubrimiento de reglas de predicción en bases de datos, reglas que se utilizarán en la mejora de Cursos Hipermedia Adaptativos basados en Web. Para la realización de la búsqueda de reglas de predicción se ha utilizado Programación Genética Basada en Gramáticas (GBGP) con técnicas de optimización multiobjetivo.

Este estudio propone como Técnica la aplicación de Algoritmos Evolutivos para llevar a cabo la tarea de extracción de conocimiento, mediante descubrimiento de reglas de predicción, ha trabajado en el paradigma de la Programación Genética Basada en Gramáticas, representando cada regla mediante un árbol de derivación de una gramática de contexto libre. Un análisis de las distintas métricas existentes para valorar la calidad de las reglas producidas, revela la necesidad de la aplicación de algoritmos multiobjetivo. En concreto, se han utilizado las aproximaciones de Algoritmos Genéticos MOGA y NSGA.

Los resultados, en función del número de reglas obtenidas, tiempo empleado en la ejecución del algoritmo, y el grado de interés, precisión y comprensibilidad de las reglas, son muy superiores y aceptables.

Como Aporte para el presente trabajo de investigación se analizara el trabajo de los Algoritmos Genéticos Multiobjetivo MOGA y NSGA usados para la generación de reglas de predicción.

(Li, Y., Wang, R., Xu, J., Yang, G., & Zhao, B., 2009), en este trabajo se describe la eficiencia del funcionamiento de los Modelos Ocultos de Markov y su gran uso en el reconocimiento de patrones, sin embargo debido al alto porcentaje de falsas alarmas en los clásicos sistemas de detección de intrusos basados en HMM, propone una técnica de enfoque borroso para el HMM, llamado HMM borroso, donde se introduce la lógica borrosa al HMM, originando que el porcentaje de robustez y precisión de los IDS basados en HMM borrosos sean más eficientes. Este método es eficiente ya que permite clasificar entre perfil anómalo y perfil normal, teniendo bajo porcentaje de falsos positivos y un gran porcentaje de aciertos.

2.3 Bases Teórico Científicas

Se presentan los conocimientos o bases teóricas que serán empleadas a lo largo de la investigación. Se considera en primer lugar las definiciones respecto seguridad de redes, sistema de respuesta a intrusos, técnicas de respuesta, sistemas autónomos de respuesta a intrusiones. Luego, se definen conceptos relacionados a los Modelos Ocultos de Markov, su aplicación y arquitectura, además de la Programación Evolutiva como herramienta para el entrenamiento de los Modelos Ocultos de Markov.

2.3.1 Seguridad en Redes

La seguridad informática es un conjunto de actuaciones que permiten proteger un entorno, asegurando de esta manera que los recursos del sistema de información de una institución sean utilizados de la forma acordada según las bases y políticas de la organización, así como que el acceso y modificación de la información contenida sólo esté permitido a personas acreditadas y dentro de unos límites establecidos.

Las actuaciones más importantes son (VILLAGRÁ GÓNZALES, 2009):

Defensa: permite disminuir la probabilidad de incidentes lesivos, es decir, intenta evitar la llegada de ataques a la organización protegida.

Seguro: actuación cuyo fin es disminuir el impacto de un incidente lesivo por si se produce la intrusión. Un ejemplo de seguro sería hacer copias de seguridad de la información.

Por otra parte, se puede entender la seguridad informática como un estado del sistema informático que indica que dicho sistema está libre de peligro, daño o riesgo. Si bien es cierto que para la mayoría de los expertos en el dominio de la seguridad informática, es utópico pensar que se pueda tener un sistema seguro 100%, puesto que día a día surgen nuevas amenazas más sofisticadas y para las que los mecanismos de seguridad aún no están preparados. Para que un sistema de información pueda definirse como seguro debe tener una serie de características:

Integridad: La información sólo puede ser modificada por personas autorizadas.



Confidencialidad: La información sólo debe ser legible para los autorizados.

Disponibilidad: Debe estar disponible cuando se necesita.

Autenticidad: el emisor de la información es quien dice ser, no se puede negar su autoría.

Al abordar el tema de la seguridad informática es necesario definir una serie de conceptos imprescindibles:

Activo: Todo lo que posee una organización susceptible de ser atacado.

Ejemplo: recursos físicos, información almacenada, utilización de los servicios/recursos, personal, información en tránsito, imagen, etc.

Amenaza: evento que puede desencadenar un incidente en la organización, produciendo daños en sus activos. Se clasifican según distintos criterios, (a) según su origen (entorno o personal); (b) según el objetivo de la amenaza (ataque a recurso físico, ataques a utilización de los recursos, ataques a la información almacenada, ataques a información en tránsito, ataques a personas o ataques a la imagen o reputación).

Vulnerabilidad: posibilidad de que una amenaza se materialice sobre un activo. Según las medidas de seguridad que posea una organización y su plan de seguridad, se es más o menos vulnerable.

Impacto: Consecuencia de la materialización de una amenaza sobre un activo. Se utilizan distintas métricas para medir el impacto, económicas o subjetivas.



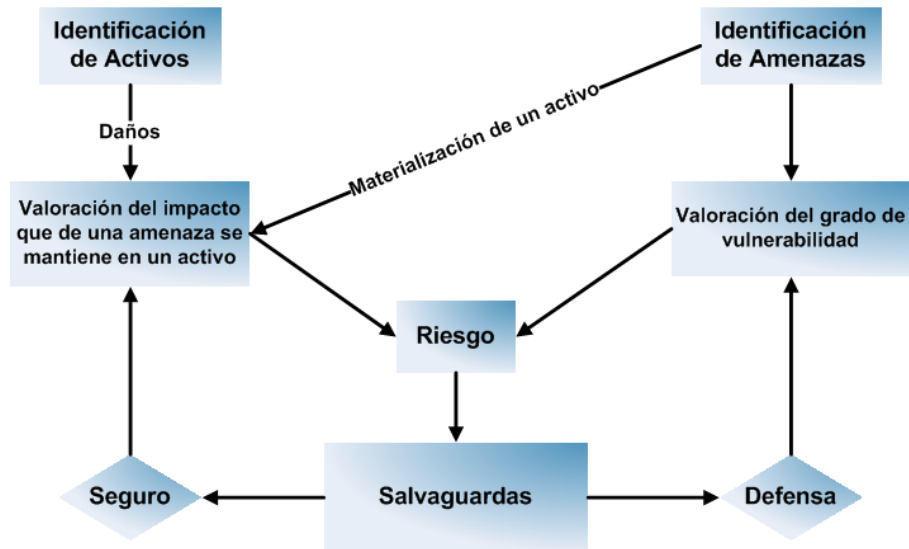
Riesgo: combinación de un cierto impacto y una vulnerabilidad. Es la posibilidad de que se produzca un impacto en un activo. El mayor riesgo es aquel al que la organización es muy vulnerable y además tiene mucho impacto para ella.

Salvaguarda: Medidas destinadas a mitigar la importancia de los riesgos. Son sistemas de prevención de ataques. Hay dos tipos, de Seguro que son medidas curativas, o de Defensa, que son medidas preventivas.

Ataque: evento, exitoso o no, que atenta sobre el buen funcionamiento del sistema.

Si se quiere dotar de gran seguridad a una organización es necesario llevar a cabo un proceso de planificación de seguridad, completa y ordenada. Esta planificación consiste en indicar de qué y cómo se quiere proteger una organización y cuánto cuesta, para lo que es necesario llevar a cabo un análisis de riesgos y de costes exhaustivo y establecer unas políticas de seguridad adecuadas. La planificación de la seguridad se basa en el Ciclo de seguridad, que se presenta a continuación:

Figura 1. Ciclo de Seguridad del Proceso de Seguridad.



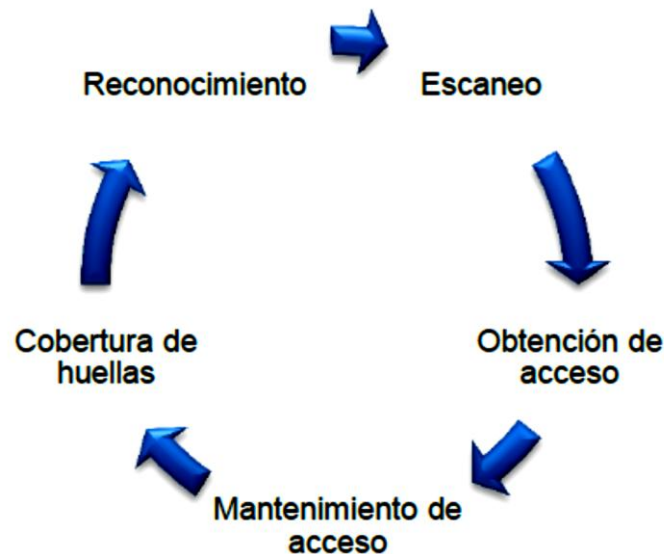
Fuente (VILLAGRÁ GÓNZALES, 2009)

2.3.2 Metodología de Intrusión

El EC-Council ha identificado una metodología estándar mediante la cual los atacantes de redes y sistemas de información realizan sus intrusiones, ésta contempla un conjunto formal de pasos para obtener y mantener el acceso a los sistemas sin importar cuales son las intenciones del intruso, la metodología ha demostrado ser muy buena, de ahí que ha sido aceptada y adoptada sin mayores variaciones por entidades como (SANS) y (EC-Council), (GRAVES , 2007), (INTERNATIONAL COUNCIL OF ELECTRONIC COMMERCE CONSULTANTS)

La metodología abarca 5 fases diferentes, las cuales definen completamente la anatomía de un ataque, que se ejecutan de manera secuencial: Reconocimiento, escaneo, obtención de acceso, mantenimiento de acceso y cobertura de huellas.

Figura 2. Fases de la Metodología de la Intrusión.



Fuente. (EC-Council)

Cada una de las fases es definida de la siguiente manera:

1. Reconocimiento: Es la etapa en la cual un atacante colecta tanta información como le sea posible con el fin de realizar una evaluación previa del blanco antes de lanzar un ataque, representa un riesgo considerable de negocio dado que la información que se busca es aquella que está presente públicamente tal como nombres de personas claves de las organizaciones, ofertas de trabajo, noticias, etc.

2. Escaneo: Esta etapa es considerada la fase previa de ataque, consiste en el sondeo de la red del blanco a partir de información recolectada durante la fase de reconocimiento, para tal fin se utilizan herramientas que son capaces de localizar sistemas activos y detectar vulnerabilidades en los mismos. En esta etapa un atacante puede obtener el mapa completo



de la red, incluyendo información sobre enrutadores, firewalls, sistemas operativos, servicios activos, versiones de software y de firmware.

3. Obtención de acceso: Esta fase es la más importante y crítica de un ataque a un sistema en términos del daño potencial que se puede causar, es el momento en el que el atacante logra tener acceso a los sistemas y tomar control de los mismos. A partir de este acceso el atacante puede robar, alterar, borrar, insertar o dañar información dentro de los sistemas que han sido víctimas. La fase consiste en la utilización de alguna técnica que le permita al atacante aprovecharse de los sistemas con el fin de obtener permisos para los cuales no está autorizado, habitualmente sucede con sistemas mal configurados, versiones viejas de aplicativos o que no han sido correctamente parcheadas, aunque también es común que el ataque sea sólo uno que evite que el sistema sea accedido por otras personas, lo que comúnmente se denomina negación de servicio.

4. Mantenimiento de acceso: Ya obtenido el acceso a los sistemas el atacante puede decidir seguir usando la víctima con el fin de lanzar más ataques o poder acceder a más información tiempo después, con tal fin se pueden implantar programas maliciosos que permiten al atacante volver a entrar al sistema sin necesidad de repetir las fases anteriores, herramientas como los caballos de Troya o las Puertas traseras son muy comunes en esta fase.

5. Cobertura de huellas: La fase final de un ataque, consiste en el borrado sistemático de evidencia que pueda incriminar o identificar al atacante que consiguió acceso al sistema. De igual manera, al cubrir un ataque el

atacante puede convertir esta fase en el inicio de un nuevo ciclo de ataque usando técnicas de reconocimiento desde el anonimato.

La primera fase no es intrusiva y prácticamente indetectable, de modo tal que las organizaciones no pueden hacer mucho para protegerse al respecto más allá de cuidar la información que publican en internet y en las entidades que registran dominios y servicios. Las últimas 2 fases radican su funcionamiento en los sistemas mismos y tienden a ser contrarrestadas con antivirus y antispyware.

Por otra parte, las fases intermedias 2 y 3 son muy ruidosas y afectan directamente las redes y sistemas de las víctimas y es allí donde las organizaciones deben comenzar su protección, siendo estrictos en la metodología, si se logra impedir que se realicen las labores de escaneo de los sistemas es imposible llevar a cabo las fases siguientes logrando desarticular el ataque antes de que logre afectar directamente los sistemas, por este motivo el presente trabajo se centra en el conjunto de ataques de estas fases, particularmente de la fase 2, y la forma de detectarlos, partiendo del hecho de que estos viajan por las redes y pueden ser capturados en la infraestructura de las víctimas justo antes de que alcancen los sistemas críticos. Para tal fin el (EC-Council) recomienda el uso de Sistemas de Detección de Intrusos como línea principal de defensa de las redes para identificar el escaneo y algunas actividades de la fase de obtención de acceso (en especial si se realizan negaciones de servicio). Del mismo modo (Institute for Security and Open Methodolog) define como factor crítico a probar la Seguridad en las tecnologías de



Internet, la cual incluye los sondeos de red y la identificación de sistemas como puntos clave, motivo adicional que muestra la importancia de la seguridad en las redes y la ventaja que puede ofrecer para prevenir ataques.

2.3.3 Tipos de Ataques Informáticos y Acciones de Respuesta Asociadas

Esta información tomada de (Mateos Lanchas, 2013), se definen los principales tipos de ataques e intrusiones actuales, explicando brevemente en qué consiste cada uno de ellos y qué vulnerabilidad o debilidades explota. Además, se indican algunas técnicas para detectarlos y un conjunto de acciones de respuesta eficaces para mitigar cada uno de los tipos de intrusión descritos.

A. Information gathering attack

Los ataques de escaneo, más conocidos como Information Gathering Attack, tienen como objetivo recopilar información sobre posibles puertas de acceso a la red. Su mecanismo de funcionamiento se basa en recopilar información que les indique qué puertos están escuchando en la red para acceder posteriormente a los recursos de la misma a través de ellos.

Dentro de los ataques de este tipo se distinguen diferentes subtipos, en función de la puerta de acceso que se escanee.

A.1 Scanning

“El escaneo como método de descubrimiento de canales de comunicación susceptibles de ser explotados, lleva en uso mucho tiempo. La idea es recorrer tantos puertos de escucha como sea posible, y guardar información de aquellos que estén receptivos y constituyan un fácil acceso al interior del sistema”.

En función de la técnica, los puertos y protocolos explotados, existen diferentes tipos de scanning. El más conocido sin duda, es el TCP scanning, forma básica del escaneo de puertos TCP.

Mecanismo de funcionamiento

Este ataque aprovecha las vulnerabilidades del protocolo TCP/IP, que utiliza puertos para envío y recepción de datos siguiendo el modelo cliente/servidor. Para establecer, mantener y terminar una conexión se utilizan bits de control (SYN, ACK, FIN) que pasan desapercibidos, vulnerabilidad que es aprovechada por los atacantes, ya que los filtros no pueden determinar con qué fin se lanzan los bits de control.

Tipos:

TCP SYN scanning: averigua puertos TCP abiertos. El atacante envía un paquete SYN y espera la respuesta. Al recibir un SYN/ACK envía inmediatamente un RST para terminar la conexión y registra ese puerto como abierto.

TCP FIN scanning: averigua puertos TCP cerrados. El atacante envía un paquete de FIN y espera respuesta. Si recibe un RST el puerto está cerrado; si no recibe respuesta, el puerteo está abierto.



Por razones de implementación del protocolo en el sistema Windows es el único que responde siempre a todas las peticiones, estén los puertos abiertos o cerrado, por lo cual no da pistas sobre que puertos hay abiertos. Los sistemas vulnerables son UNIX LINUX NOVELL.

Las principales ventajas de esta técnica de ataque es que no necesita privilegios especiales y posee una gran velocidad.

Su principal desventaja es la facilidad de detección.

Detección

Este tipo de ataques se detecta por el gran número de conexiones y mensajes de error que se observan para los servicios a los que se ha conseguido conectar el atacante e inmediatamente se ha desconectado, ya que el ataque TCP scanning nunca abre una sesión TCP completa.

Acciones de respuesta

Como acciones de respuesta se pueden llevar a cabo las siguientes:

-Monitorizar la red en busca de paquetes SYN a puertos restringidos (Respuesta Pasiva y Respuesta Proactiva).

-Bloqueo de mensajes salientes con origen el puerto X (puerto sobre el que se han detectado las conexiones y mensajes de error) al puerto Y (el del atacante; el puerto origen de los paquetes SYN). (Respuesta Activa de Protección y Respuesta Proactiva).

-Desactivar/Desinstalar servicios innecesarios. (Respuesta Proactiva).

Cambiar el puerto a uno más alto, para evitar los escaneos (Respuesta

-Activa de Protección y Respuesta Proactiva).

-Rastrear la conexión del atacante para recopilar información (Respuesta Activa de Reacción).

A.2 Sniffing

Ataque pasivo cuyo objetivo es la interceptar tráfico de red de manera pasiva, sin modificación del mismo.

Mecanismo de funcionamiento

Las herramientas utilizadas para llevar a cabo el ataque son los denominados Packet Sniffers, programas que monitorizan los paquetes que circulan por la red, que pueden instalarse en cualquier equipo de la red.

El ataque consiste en colocar la tarjeta de red en modo “promiscuo” que desactiva el filtro de verificación de direcciones lo que provoca que todos los paquetes enviados a la red lleguen a esta placa (y pueden ser analizados por el atacante).

Actualmente existen Sniffers que mediante el análisis del tráfico capturado pueden capturar todo tipo de información específica, como passwords de un recurso compartido o de acceso a una cuenta, números de tarjetas de crédito, direcciones de e-mails entrantes y salientes, relaciones entre individuos, etc.

Detección

Para su detección, es recomendable realizar las siguientes acciones:

Revisar la lista de programas en ejecución para detectar alguna anomalía.

- Mirar la lista de programas que se inician automáticamente al encender el PC o las tareas programadas para buscar evidencias.

- Detectar tarjetas o adaptadores de red en modo promiscuo mediante el ipconfig. (el modo promiscuo es el usado por los atacantes para instalar los sniffers).

Acciones de respuesta

Como acciones de respuesta se pueden llevar a cabo las siguientes:

- Cerrar el puerto X de la máquina atacada al que el atacante puede conectarse para recuperar los datos que el sniffer ha capturado, es decir, interrumpir la conexión de red (Respuesta Activa de Protección y Respuesta Proactiva).

- Matar el proceso resultado de la ejecución del sniffer. (Respuesta Activa de Protección).

- Eliminar el fichero donde se están grabando los datos, imposibilitando así su acceso por parte del atacante (Respuesta Activa de Restauración).

NOTA: estas tres respuestas se pueden llevar a cabo si el sniffer no se ha ocultado de forma adecuada y es detectado por los IDSs del sistema, caso en el que las herramientas de monitorización muestran el proceso que se está ejecutando, el fichero donde se están grabando los datos y la conexión de red, lo que permitirá ejecutar la respuesta adecuada.

Un ataque muy similar a sniffing pero que además de interceptar el tráfico de red, el atacante se descarga la información capturada a su propia máquina para analizarla de forma más exhaustiva, es el conocido como Snooping. Las acciones de detección y respuesta ante este tipo de ataque son idénticas a las indicadas para sniffing.

B. Network attacks (Ataques de red)

Ataque cuyo objetivo es introducirse en un sistema suplantando la identidad de un usuario o del propio administrador, mediante la manipulación de protocolos de la torre de protocolos TCP/IP, aprovechando una sesión establecida por el usuario víctima, o utilizando los datos obtenidos mediante ataques de escaneo (nombre de usuario y password).

Este ataque a su vez se divide en 4 tipos: Hijacking, Spoofing, ataques Wireless y ataques a aplicaciones Web o Web Application Attacks, que a continuación se describen. Este último tipo tiene especial relevancia.

B.1 Spoofing

Ataque consistente en que el atacante intenta ganar acceso a un sistema haciéndose pasar un usuario que dispone de los privilegios suficientes para realizar la conexión.

Definición. Mecanismo de funcionamiento

Para la realización de este ataque se utilizan técnicas de suplantación de identidad. Además, para llevarlo a cabo se necesita poseer gran conocimiento del protocolo en el que se basa el ataque. En función de la entidad objetivo de la suplantación y el protocolo, existen diferentes tipos:

- IP-Spoofing: suplantación de IP. Consiste en sustituir la dirección IP origen de un paquete TCP/IP por la IP que se quiere suplantar; las respuestas irán dirigidas a la IP falsificada. El atacante genera paquetes (TCP/IP) con una dirección IP de destino que no es la de la

máquina víctima, sino una de una maquina determinada que sostiene una "relación de confianza" con el objetivo.

DNS-Spoofing: suplantación de identidad por nombre de dominio. Consiste en suplantar el DNS legítimo y falsificar la relación “nombre de dominio – IP”, de tal forma que cualquier consulta de resolución de nombre se resuelva con una dirección IP falsa de un cierto dominio DNS o viceversa. Esto se consigue falseando las entradas de la relación Nombre de dominio-IP de un servidor DNS, mediante alguna vulnerabilidad del servidor en concreto o por su confianza hacia servidores poco fiables. De esta manera, el atacante proporciona a los servidores que realizan las consultas direcciones IP o nombres de dominio falsos con fines maliciosos, redirigiéndoles hacia páginas incorrectas. Las entradas falseadas de un servidor DNS son susceptibles de infectar (envenenar) el cache DNS de otro servidor diferente (DNS Poisoning).

- Web- Spoofing: suplantación de una página web legítima. Consiste en redirigir la conexión de la máquina víctima hacia otras páginas Web que suplantán a la original, con el objetivo de obtener información acerca de la víctima. La página web falsa actúa a modo de proxy, solicitando la información requerida por la víctima a cada servidor original. El atacante puede modificar cualquier información desde y hacia cualquier servidor que la víctima visite. La forma en la que la víctima accede a la página web suplantada, suele ser mediante ataques de phishing. Este ataque se realiza mediante una implantación

de código el cual nos robará la información, y es difícilmente detectable.

- ARP- Spoofing: suplantación de identidad mediante falsificación de tablas ARP. El ataque consiste en construir tramas de solicitud y respuesta ARP modificadas con el fin de falsear la tabla ARP (relación IP-MAC) haciendo que la máquina víctima envíe los paquetes a un host atacante en lugar de a su destino elegido. El protocolo ARP trabaja a nivel de enlace de datos de la torre OSI, por lo que esta técnica sólo puede ser utilizada en redes LAN o en cualquier caso en la parte de la red que queda antes del primer router.

- TCP-Spoofing: ataque spoofing más común, que consiste en falsificar una conexión TCP. Se conoce como adivinación del número de secuencia, y se basa en la idea de que si un atacante puede predecir el número inicial de secuencia de la conexión TCP generado por la máquina destino, el atacante puede adoptar la identidad de máquina de confianza y establecer una conexión con la máquina destino.

Los ataques de spoofing afectan a diario el funcionamiento de las redes TCP/IP. Estos son empleados como parte de un ataque o individualmente. La sencillez de su explotación ha masificado su uso.

Todos los servicios TCP/IP sufren estas afectaciones.

Detección

Es un ataque difícil de detectar y de evitar.

En el caso de Web spoofing, quizá la mejor medida sea instalar algún plugin en el navegador que muestre en todo momento la IP del servidor



visitado: si la IP nunca cambia al visitar diferentes páginas WEB significará que probablemente estemos sufriendo este tipo de ataque.

Acciones de respuestas

Como acciones de respuesta ante este tipo de ataques se pueden llevar a cabo:

- Configurar el router para que no admita el envío de paquetes con IP origen no perteneciente a una de las redes que administra. Filtrado de paquetes por IP origen (Respuesta Activa de Protección y Respuesta Proactiva).
- Filtrar direcciones en el cortafuego (Respuesta Activa de Protección y Respuesta Proactiva).
- Realización de backup periódicos de la información sensible de los sistemas, que podrá ser utilizada tras la consecución de un ataque como respuesta de recuperación. (Respuesta Activa de Recuperación y Respuesta Proactiva).

Estrategia antispoofting

Una buena estrategia antispoofting, es aquella que propone considerar todos los riesgos que coexisten en la red TCP/IP para protegerla contra un ataque de spoofing, como respuesta de prevención, teniendo en cuenta que todos los servicios esenciales son vulnerables al spoofing y que la protección es similar para todos los casos.

Entre las acciones de prevención, protección y recuperación que constituyen la estrategia se encuentran:

- Filtrado de paquetes en el router externo del cortafuegos. Es imposible eliminar todos los paquetes que tengas direcciones falsas

(spoofing IP), pero sí es posible reducir el número si se filtran los paquetes y se restringe el flujo de entrada y salida de la red.

- Configuración segura de los servicios DNS, SMTP, etc.
- Protección de las conexiones y los datos, para mantener la confidencialidad, autenticidad e integridad de la información. Por ejemplo, cifrar las conexiones o utilizar la firma digital.
- Utilizar barreras o filtros de red para registrar todas las conexiones a un servidor determinado.
- Utilizar sistemas proxies para la navegación. Con esto se pretende detectar los posibles ataques de spoofing WWW.
- Instalar programas sniffers que permiten detectar la ocurrencia de algunos ataques.
- Explotar de forma adecuada los logs de los servicios y sistemas operativos. La configuración de la generación de logs debe estar encaminada a almacenar la mayor cantidad de información útil posible. Debe registrarse por ejemplo: dirección IP origen de las conexiones, el nombre del usuario, horario, duración de la conexión y tamaño e identificación de la transferencia efectuada.
- Educar a los usuarios, que representa una forma de defensa muy importante contra el spoofing WWW.
- Segmentación del tráfico, con el objetivo de separar todo el tráfico de administración de los sistemas del de los usuarios comunes y del de Internet.
- Utilizar herramientas de detección de vulnerabilidades.
- Actualizar los parches de los sistemas operativos.

- Ofrecer sólo los servicios imprescindibles y de manera segura.

B.2 Hijacking

Mecanismo de funcionamiento

Ataque que consiste en ocupar una identidad obteniendo acceso no autorizado a un sistema, cuenta, etc., con el objetivo de poder falsificar la identidad del usuario atacado. Este tipo de ataque se produce cuando el atacante consigue interceptar una sesión ya establecida. Es un ataque de apropiación de identidad, que atenta contra la autenticación de los usuarios. El atacante roba una conexión después de que el usuario ha superado con éxito el proceso de identificación ante el sistema. El ordenador desde el que se lanza el ataque ha de estar en alguna de las dos redes extremo de la conexión, o en la ruta entre ambas.

Detección

Es un ataque muy difícil de detectar.

Acciones de respuesta

Como acciones de respuesta que pueda desplegar un AIRS de forma automática no hay ninguna efectiva. En cuanto a recomendaciones para evitar este tipo de ataques, se recomienda cifrar la información. Es el método más seguro para protegerse contra este tipo de ataques.

C. Web application Attacks (Ataques a aplicaciones web)

Los ataques contra aplicaciones web se han convertido en una de las amenazas más serias para las infraestructuras de seguridad informática, ya que ponen en peligro la información corporativa y los datos confidenciales de los clientes. Las aplicaciones web

desplegadas en la parte pública de internet atraen ataques de hackers y gusanos, constituyendo a menudo el punto más vulnerable de las infraestructuras de red. Por ello, es imprescindible implantar fuertes medidas de seguridad a nivel de transacciones HTTP, HTTPS y FTP, ya que de lo contrario pueden convertirse en punto de entrada a las redes corporativas, desde las cuales robar información confidencial o atacar otros servidores internos.

Mecanismo de funcionamiento

Hay gran cantidad de ataques a aplicaciones web; a continuación se definen brevemente los más comunes:

- Ejecución de código de forma remota: el atacante ejecuta código en el servidor vulnerable y obtiene información almacenada en él. Este ataque explota las vulnerabilidades del sistema debidas a errores de programación. Es un ataque simple pero eficaz.
- Inyección de código SQL o SQL Inyencion (ataques a bases de datos): ataque que afecta directamente a las bases de datos de una aplicación. Este ataque consiste en insertar o inyectar código SQL malicioso dentro de código SQL, para alterar el funcionamiento normal y hacer que se ejecute el código “invasor” dentro del sistema atacado. Esta técnica de ataque se usa para explotar sitios web que construyen sentencias SQL a partir de entradas facilitadas por el usuario. Es muy común entre los atacantes y dependiendo de las medidas de seguridad que tenga el sistema, el impacto del ataque puede variar desde revelación de información básica a ejecución de código remoto y

compromiso total del sistema, mediante la alteración parcial o total de las bases de datos del sistema comprometido.

- Cross Site Scripting (XSS): ataque que consiste en explotar las vulnerabilidades del sistema de validación de HTML incrustado, mediante la inyección en páginas web de código JavaScript o en otro lenguaje script similar con fines maliciosos. Es posible encontrar una vulnerabilidad XSS en aplicaciones que tenga entre sus funciones presentar la información en un navegador web u otro contenedor de páginas web. Sin embargo, no se limita a sitios web disponibles en Internet, ya que puede haber aplicaciones locales vulnerables a XSS, o incluso el navegador en sí.

XSS es un vector de ataque que puede ser utilizado para robar información delicada, secuestrar sesiones de usuario, y comprometer el navegador, subyugando la integridad del sistema. El uso de AJAX para ataques de XSS no es tan conocido, pero sí peligroso. Se basa en usar cualquier tipo de vulnerabilidad de XSS para introducir un objeto XMLHttpRequest y usarlo para enviar contenido POST, GET, sin conocimiento del usuario.

- Envenenamiento de cookies: ataque que consiste en la modificación del valor de las cookies antes de enviarlas de vuelta al servidor. A día de hoy, la mayoría de los sitios web, sólo almacenan un identificador de sesión (un número único generado aleatoriamente usado para identificar la sesión de usuario) en la propia cookie, mientras que el resto de la información es almacenada en el servidor, eliminando en gran medida el riesgo de sufrir este tipo de ataques.



Suplantación de contenido: técnica de ataque utilizada para engañar al usuario haciéndole creer que cierto contenido que aparece en un sitio web es legítimo, cuando en realidad no lo es.

- Otros ataques definidos en otras partes del documento: ataques de fuerza bruta, desbordamiento de buffer, denegación de servicio, etc. Hay gran cantidad de ataques contra aplicaciones Web.

Acciones de respuesta

Entre las acciones de respuesta que pueden llevarse a cabo ante este tipo de ataque se encuentran:

- Utilizar cortafuegos de aplicaciones web (WAF) ya que los cortafuegos tradicionales trabajan en la capa de red y de transporte, y al dejar el puerto 80 abierto no ofrecen ninguna clase de protección frente a los ataques contra aplicaciones web. Los WAF pueden ser a nivel de host y a nivel de red (Respuesta Activa de Protección y Respuesta Proactiva).
- Reconfigurar o restaurar el cortafuegos de aplicación (Respuesta Activa de Protección y Respuesta Activa de Recuperación).
- Evitar conexiones a bases de datos como un superusuario o como administrador de la base de datos (Respuesta Activa de Protección y Respuesta Proactiva).
- Cerrar el servicio o aplicación que proporciona el servidor afectado (Respuesta Activa de Protección).
- Reiniciar el servicio o aplicación que proporciona el servidor afectado (Respuesta Activa de Protección y Respuesta Activa de Recuperación).

- Restaurar el contenido de una página web en caso de que el servidor web sea el objetivo del ataque (Respuesta Activa de Recuperación).

D. BackDoors

Una puerta trasera (backDoor) en sí es un programa que permite remotamente acceder a un sistema con libertad de movimientos, que constituye un acceso a un sistema o programa de forma transparente al usuario.

Mecanismo de funcionamiento

Es una técnica de ataque que consiste en la utilización de trozos de código en un programa que permite a quien las conoce saltarse los métodos usuales de autenticación para realizar ciertas tareas.

Puede establecerse por dos tipos de usuarios: un programador que desarrolla una aplicación y establece backdoors para acceder al programa más rápidamente en la fase de pruebas o por error; un atacante que establece una backdoor mediante otras técnicas de ataque como exploits o troyanos para tener acceso a la máquina víctima con fines maliciosos.

Una puerta trasera no es peligrosa si es utilizada por usuarios no malintencionados, pero es una entrada al sistema no controlada, lo que la convierte en peligrosa para la seguridad e integridad del sistema.

La mayoría de las puertas traseras se introducen en nuestro sistema en forma de troyanos cuya finalidad es siempre malévol. Por otra parte, para poder utilizar las puertas traseras debe existir un cliente y un servidor, pero los troyanos tienen la capacidad de instalar en el

sistema un servidor para posteriormente acceder a él como cliente y tener control remoto.

El hecho de tener una entrada no controlada al sistema, provoca un fallo de seguridad si se utiliza de forma malintencionada, ya que cualquiera que conozca el agujero o lo encuentre en su código podrá saltarse los mecanismos de control y autenticación habituales.

Para abrir una puerta trasera en un sistema Unix a veces basta añadir dos líneas de texto.

Detección

Para detectar una backDoor hay que tener en cuenta los efectos que produce la instalación de ésta en el sistema. Por ejemplo, se puede saber que hay instalada una backdoor si se detecta la siguiente actividad no autorizada por el usuario o administrador del sistema:

- Monitorización del sistema completo
- Descarga de virus.
- Desinstalación e instalación de programas.
- Borrado de información de usuario.
- Manipulación del registro del sistema.

Acciones de respuesta

Entre las acciones de respuesta que pueden llevarse a cabo ante este tipo de ataque se encuentran:

- No aceptar peticiones ni llamadas que provengan de usuarios desconocidos o sospechosos (Respuesta Activa de Protección y Respuesta Proactiva).

- En el caso de borrado de información de usuario, restaurar el activo afectado a su versión anterior previa al ataque. (Respuesta Activa de Recuperación).
- Aumentar la sensibilidad del IDS para la detección de troyanos (Respuesta Activa de Recuperación, Respuesta Pasiva y Respuesta Proactiva).
- Restaurar el sistema a una versión correcta previa a la intrusión (Respuesta Activa de Recuperación).
- Identificar la backdoor y bloquearla, cerrarla (Respuesta Activa de Protección y Respuesta Proactiva).

E. Password Attacks (Ataques a contraseñas)

Un porcentaje elevado de intrusiones en un sistema se deben al fallo del sistema de autenticación. Todo sistema depende en su mayoría del ingreso de un password o contraseña, técnica muy vulnerable. La utilización de password como método de autenticación conlleva una serie de problemas: se olvidan, se comparten con personas de confianza que luego podrían resultar no tan confiables, se interceptan o escuchan por usuarios ajenos, etc. Existen otros mecanismos de autenticación y autorización más seguros y robustos que la autenticación mediante contraseñas, como por ejemplo, los mecanismos de autenticación biométrica (autenticación por huellas digitales, iris, forma de la cara, etc.). Estos sistemas presentan una serie de inconvenientes, como es la necesidad de disponer de periféricos adecuados en los sistemas informáticos remotos que permitan capturar la información biométrica necesaria.



Además de los ataques de ingeniería social (ataques mediante engaño del usuario) existen dos técnicas para ejecutar este tipo de ataque.

E.1 Guessing. Brute Force

Ataque que consiste en generar e ir probando con todas las combinaciones de caracteres (letras, números, etc.) posibles hasta adivinar la contraseña (password). Es un proceso automatizado de prueba y error.

Mecanismo de funcionamiento

Este tipo de técnica consiste en la obtención por "fuerza bruta" de aquellas claves que permiten acceder a los sistemas, aplicaciones, cuentas, etc. objetivos del ataque. Esta técnica es altamente eficaz ya que la mayoría de las contraseñas de acceso se pueden obtener fácilmente debido a la mala concienciación de los usuarios, que suelen elegir contraseñas fáciles, cortas y no las cambian con regularidad.

Un ataque de fuerza bruta "pura" teóricamente no puede ser resistido por ningún sistema, siempre y cuando se disponga del tiempo suficiente y recursos suficientes. Será más complicado adivinar claves largas, pero si no existe limitación de tiempo, el atacante conseguirá su objetivo. Estas limitaciones físicas son dinámicas y van disminuyendo a medida que los ordenadores van alcanzando más capacidad de procesamiento.

Como medida de protección, muchos sistemas niegan el acceso después de un determinado número de fallos de autenticación, lo que dificulta la labor de los atacantes que usan esta técnica.

Por otra parte, esta técnica tiene la ventaja de que la mayoría de los ataques de fuerza bruta se hacen offline, es decir, se obtiene el archivo que contiene la lista cifrada de passwords y se trabaja desde la propia máquina del atacante, sin necesidad de mantener conexión con la máquina víctima.

Detección

Como técnicas de detección de ataques de adivinación por fuera bruta se tienen:

- Instalar en los sistemas de la organización alguna herramienta de detección de ataques de fuerza bruta, como por ejemplo Denyhosts, blocksshd, fail2ban, Brute Force Detection, etc., que informan en cada momento de los intentos no autorizados y todo tipo de actividad de autenticación sospechosa que se ejecute contra los servidores, en especial contra el servidor SSH.
- Mirar los logs de autenticación del sistema.
- Ataque fácil de detectar por la mayoría de los cortafuegos (un gran número de intentos de conexión desde una sola dirección IP). No obstante, desde la utilización de redes botnets para la masificación de ataques, es más difícil su detección.

Acciones de respuesta

Entre las acciones de respuesta que pueden llevarse a cabo ante este tipo de ataque se encuentran:

- Añadir una entrada que deniegue el acceso al usuario atacante detectado, (Respuesta Activa de Protección y Respuesta Proactiva).

- Añadir reglas a la configuración de las tablas IP de la máquina atacada que impidan cualquier tipo de tráfico entre el atacante y la máquina atacada. Bloquear la IP o servicio implicado. (Respuesta Activa de Protección y Respuesta Proactiva).
- Hacer un whois de la IP y rastrear la conexión del atacante para recopilar información (Respuesta Activa de Reacción).
- No permitir acceso a root en sistemas Linux (Respuesta Activa de Protección y Respuesta Proactiva).
- Restaurar los activos afectados (modificados, eliminados, etc.) con motivo de la intrusión (Respuesta Activa de Recuperación).

E.2 Guessing. Dictionary attack

Ataque similar al anterior, que consiste en probar con todas (o la mayoría) de las palabras conocidas en un idioma dado: un buen diccionario tiene entre 100 mil y 200 mil palabras y es un excelente punto de partida. No es lo mismo que un ataque de pura fuerza bruta, que prueba todas las combinaciones posibles.

Mecanismo de funcionamiento

Como se indica en, “Los diccionarios son archivos con millones de palabras, las cuales pueden ser passwords utilizadas por los usuarios”.

El atacante utiliza un programa que se encarga de probar cada una de las palabras cifrando cada una de ellas (mediante el algoritmo utilizado por el sistema atacado) y comparando la palabra cifrada contra el archivo de passwords del sistema atacado (previamente obtenido). Si coinciden se ha encontrado la clave de acceso al sistema mediante el usuario correspondiente a la clave hallada. Actualmente es posible

encontrar diccionarios de gran tamaño orientados, incluso, a un área específica de acuerdo al tipo de organización que se esté atacando.

Los ataques de diccionario tienen pocas probabilidades de éxito con sistemas que utilizan contraseñas fuertes con letras en mayúsculas y minúsculas mezcladas con números y con cualquier otro tipo de símbolos. Sin embargo para la mayoría de usuarios recordar contraseñas tan complejas resulta complicado, por lo que un ataque de diccionario suele ser más eficaz que uno de fuerza bruta.

Detección

Los mecanismos de detección de esta técnica de ataque de adivinación son similares a los especificados para Guessin - Brute Force.

Acciones de respuesta

Las acciones de respuesta de esta técnica de ataque de adivinación son similares a los especificados para Guessin - Brute Force.

F. Exploits

Ataque que consiste en utilizar un trozo de código, fragmento de datos o secuencia de comandos y/o acciones llamado exploit, para explotar los agujeros o vulnerabilidades de seguridad existentes en un sistema de información, con el objetivo de conseguir un acceso no autorizado a la máquina objetivo o provocar un comportamiento no deseado de dicho sistema o de alguno de los servicios que éste presta. Los objetivos perseguidos por los atacantes mediante la explotación de exploits son muchos y variados, como por ejemplo, conseguir acceso

al sistema de forma no autorizada, obtener privilegios no concedidos de forma lícita, realizar un ataque de denegación de servicio, etc.

Cabe aclarar que el término exploit hace referencia al trozo de código que el atacante ejecuta para explotar la vulnerabilidad encontrada y conseguir su objetivo.

Mecanismo de funcionamiento

Para llevar a cabo esta técnica de ataque, los atacantes ejecutan programas que permiten explotar estos “agujeros” de seguridad que reciben el nombre de Exploits. El funcionamiento es el siguiente: el programa aprovecha la debilidad, fallo o error hallado en el sistema (hardware o software) para conseguir acceso al mismo. Nuevos exploits (que explotan nuevas vulnerabilidades en los sistemas) se publican cada día por lo que mantenerse informado de los mismos y de las herramientas para combatirlos es esencial.

Detección

Como técnicas de detección de este tipo de ataques se distinguen:

Los programas detectores de exploits se centran en la detección de malware existente en el PC después de que la vulnerabilidad haya sido explotada, actualizándose a posteriori con firmas o ficheros de datos para reconocer las cargas dañinas instaladas sobre el PC tras explotar las brechas de seguridad existentes.

- Identificación del código de explotación, para lo que deben analizarse las versiones del programa afectado por la vulnerabilidad antes y después de la aplicación del parche correspondiente para averiguar cómo funciona tal código.

En máquinas Windows hay ciertos indicios que permiten detectar que un atacante ha explotado una vulnerabilidad mediante un exploit:

- El tráfico de red de salida es sospechosamente alto, sobre todo cuando el ordenador está inactivo o no necesariamente cargando datos.
- Reducción drástica del espacio libre en disco, o existencia de archivos de aspecto sospechoso en los directorios raíces de cualquiera de los discos. Después de penetrar en el sistema, muchos hackers realizan un escaneo masivo para encontrar documentos interesantes o archivos que contengan contraseñas.
- Se produce un gran aumento en el número de paquetes que son bloqueados por un cortafuegos personal, procedentes de una dirección simple.

De la misma forma, posibles indicios de exploits en máquinas Unix son:

- Existencia de archivos con nombres sospechosos en el archivo /tmp folder.
- Instalación de backdoors o modificación de utilidades estándar del sistema que se usan para conectarse con otros sistemas.
- Alteración de los ficheros /etc/passwd, /etc/shadow, u otros archivos de sistemas en el directorio /etc, por un usuario sin privilegios. A veces los atacantes pueden añadir un nuevo usuario en el fichero /etc/passwd, para que este tenga acceso remoto al sistema de forma lícita en una fecha posterior.
- Modificación del fichero /etc/services así como de /etc/inet.conf, para establecer una backdoor en puerto sospechoso o no usado.

Acciones de respuesta

Entre las acciones de respuesta que puede llevar a cabo el AIRS ante este tipo de ataque se encuentran:

- Agregar o borrar reglas de filtrado en los cortafuegos, (Respuesta Activa de Protección y Respuesta Proactiva).
- Bloquear la IP del atacante temporalmente hasta que cesen los intentos de conexión (Respuesta Activa de Protección y Respuesta Proactiva).
- Terminar conexión TCP establecida por el atacante (Respuesta Activa de Protección y Respuesta Proactiva).
- Bloquear conexiones de red entrantes y salientes sospechosas (Respuesta Activa de Protección y Respuesta Proactiva).
- Eliminar cualquier nombre de usuario sospechoso en el archivo de contraseñas (Respuesta Activa de Protección y Respuesta Proactiva).

G. Virus

Un virus es un programa que se replica a sí mismo y que se propaga a través de archivos infectados. Una vez cargado en memoria cumple las instrucciones programadas por su creador, que pueden ser: destruir ficheros, modificar ficheros, sobrecargar recursos, entre otros.

Mecanismo de funcionamiento

Un virus es un trozo de código incrustado en un programa que se replica a sí mismo y se propaga a través de archivos infectados, cuyo objetivo es alterar el funcionamiento normal de la máquina atacada, creando o modificando ficheros, sobrecargando los recursos del sistema, etc.

Un virus tiene las siguientes características:

- Se replican (propagan) y ejecutan por sí mismos.
- Reemplazan archivos ejecutables por otros infectados con el código de éste.
- Contienen una carga dañina (payload) con distintos objetivos: destruir datos dañando el sistema; bloquear las redes informáticas generando tráfico inútil; ser “algo molestos”.

Su mecanismo de funcionamiento es el siguiente:

- Se ejecuta un programa infectado (por desconocimiento del usuario).
- El código del virus queda residente en RAM, aun cuando el programa que lo contenía haya terminado.
- El virus toma el control de los servicios básicos del SO, infectando archivos ejecutables que serán llamados posteriormente.
- Finalmente, se añade el código del virus al del programa infectado y se graba en disco, con lo cual el proceso de replicación se completa.

Los efectos de un virus varían desde la simple visualización de una pelota de ping pong rebotando por toda la pantalla de escritorio de la máquina víctima, hasta la eliminación de ficheros o desinstalación de programas.

Tipos de virus

Los virus se clasifican según diferentes criterios. Se pueden clasificar, por ejemplo, según el tipo de activo infectado (ref11):

- Archivos:

Virus de acción directa: al ejecutarse, infectan otros programas.



Virus residentes: al ejecutarse, se instalan en la RAM. Infectan a los demás programas a medida que se accede a ellos.

- Sector de arranque (virus de boot). Estos virus residen en la memoria.
- Ambos: infectan archivos y al sector de arranque.

También se clasifican según su comportamiento:

- Kluggers: al entrar en otros sistemas se reproducen y cifran de manera que sólo se les puede detectar con algún tipo de patrones.
- Vidddbers: modifican los programas del sistema en el cual entran.

Así mismo, existen otras posibles clasificaciones:

- Virus uniformes: producen una replicación idéntica a sí mismos.

Virus cifrados: cifran parte de su código para que sea más complicado su análisis.

- Virus oligomórficos: usan funciones de encriptación aleatorias.

Requieren distintos patrones para su detección.

- Virus polimórficos: en su replicación producen una rutina de encriptación variable, tanto en la fórmula como en el algoritmo.

Requiere técnicas antivirus avanzadas.

- Virus metamórficos: reconstruyen todo su cuerpo en cada generación, haciendo que varíe por completo. De esta forma se llevan

las técnicas avanzadas de detección al límite.

- Sobrescritura: el virus sobrescribe a los programas infectados con su propio cuerpo.

- Stealth o silencioso: el virus oculta síntomas de la infección.



Detección

Como técnicas de detección de este tipo de ataques se distinguen:

- Uso de programas antivirus que analizan la firma del virus.
- Cambios producidos en la organización del sistema de ficheros o en la información del sistema.
- Ejecución del método heurístico, que implica el análisis del comportamiento de las aplicaciones para detectar una actividad similar a la de un virus conocido.
- Aumento del consumo de recursos del sistema que implican una pérdida de productividad, cortes en los sistemas de información o daños a nivel de datos.

Acciones de respuesta

Entre las acciones de respuesta que puede llevar a cabo el AIRS ante este tipo de ataque se encuentran:

- Analizar las trazas que ha dejado un virus para eliminarlo una vez detectado (Respuesta Activa de Reacción).
- Generar filtros de ficheros dañinos, por ejemplo en el sistema de correo.
- Realizar copias de seguridad de todos los activos del sistema (Respuesta Proactiva) con el fin de futuras restauraciones (Respuesta Activa de Recuperación).
- Restauración completa de la máquina o sistema comprometido (formateo) (Respuesta Activa de Recuperación).
- Denegar el acceso de forma selectiva o completa a un archivo (Respuesta Activa de Protección y Respuesta Proactiva).

H. Worms (Gusanos)

Un worm o gusano es un programa que se reproduce por sí mismo, que puede viajar a través de redes utilizando los mecanismos de éstas y que no requiere respaldo de software o hardware (como un disco duro, un programa host, un archivo, etc.) para difundirse.

Mecanismo de funcionamiento

Un gusano puede considerarse una técnica de ataque similar a los virus pero que no dependen de archivos portadores para poder llegar e infectar otros sistemas.

Las principales características de los gusanos son:

- No altera los archivos del sistema sino que residen en la memoria y se duplica a sí mismo.

Pueden modificar el SO con el fin de autoejecutarse como parte del proceso de inicialización del sistema.

- Utilizan las partes automáticas de un SO que generalmente son invisibles al usuario.

Principalmente su mecanismo de funcionamiento es el siguiente: explotar vulnerabilidades de la máquina objetivo o utilizar algún tipo de ingeniería social para engañar a los usuarios y poderse ejecutar. Los gusanos actuales se propagan principalmente con clientes de correo electrónico (en especial de Outlook) mediante el uso de adjuntos que contienen instrucciones para recolectar todas las direcciones de correo electrónico de la libreta de direcciones y enviar copias de ellos mismos a todos los destinatarios.

Generalmente, estos gusanos son scripts (típicamente en VBScript) o archivos

Detección

Como técnicas de detección de este tipo de ataques se distinguen:

- Aumento drástico del consumo de recursos de la máquina víctima. Tienen una incontrolada replicación, por lo que consumen muchos recursos del sistema, hasta el punto de que los procesos del sistema se ejecutan de forma excesivamente lenta o no pueden ejecutarse.
- Modificación de información del sistema de forma no autorizada.
- Ver Virus. Detección.

I. Troyanos

La intrusión conocida bajo el nombre de Troyano, consiste en un programa oculto dentro de otro que ejecuta comandos furtivamente y que, por lo general, abre el acceso al ordenador y abriendo una backdoor.

Mecanismo de funcionamiento

Este ataque se basa en la utilización de un código malicioso que se encuentra en un programa sano (por ejemplo, un comando falso para crear una lista de archivos que los destruye en lugar de mostrar la lista).

El atacante introduce el trozo de código malicioso dentro de un programa útil del sistema. La ejecución del programa origina la ejecución de la rutina, instalando así el troyano en el sistema al ejecutar dicho archivo. El objetivo de este ataque no es reproducirse para infectar a otras máquinas, sino abrir un puerto en la máquina

víctima para que un atacante pueda establecer conexión con ella a través del puerto o backdoor establecida, y tener control sobre la máquina comprometida. Para ello, debe infectar la máquina obligando a abrir un archivo infectado que contiene el Troyano, y luego, acceder a la máquina a través del puerto abierto. Hay una serie de puertos fijos utilizados generalmente por los troyanos, como por ejemplo, el puerto 28678 utilizado por Exploiter, el puerto 29104 utilizado por NetTrojan, puerto 29369 utilizado por ovasOn, etc.

Una vez instalado, el troyano puede realizar las siguientes tareas:

- Espiar al usuario: obtención de contraseñas mediante captura de las pulsaciones del teclado, etc.
- Realizar operaciones maliciosas dentro de la máquina como la instalación de programas.
- Creación de un agujero de seguridad en la red para que los usuarios externos puedan acceder a áreas protegidas de esa red.
- Establecimiento de BackDoors, para que el atacante pueda acceder a la máquina comprometida.
- Etc.

Detección

La infección es evidente por los siguientes síntomas:

- Actividad anormal de dispositivos de red o del propio sistema, como por ejemplo:

Reacciones extrañas de los dispositivos periféricos como el ratón o el teclado.

Programas que se abren en forma inesperada.

Bloqueos repetidos.

Acciones de respuesta

Entre las acciones de respuesta que puede llevar a cabo un AIRS ante este tipo de ataque se encuentran:

- Agregar o borrar reglas de filtrado sobre el cortafuegos (Respuesta Activa de Protección y Respuesta Proactiva), que bloqueen el establecimiento de conexiones salientes por parte de programas cuyo origen se desconoce (Respuesta Activa de Protección y Respuesta Proactiva).
- Eliminar el troyano, mediante herramientas como The Cleaner (Respuesta Activa de Recuperación)
- Formatear la máquina afectada (Respuesta Activa de Protección y Respuesta Proactiva).
- Cerrar el puerto utilizado por el troyano para abrir una backdoor (Respuesta Activa de Protección y Respuesta Proactiva).
- Realizar copias de seguridad de todos los ficheros del sistema (Respuesta Proactiva).
- Rastrear la conexión del atacante para recopilar información (Respuesta Activa de Reacción).
- Denegar el acceso de forma selectiva o completa a un archivo (Respuesta Activa de Protección y Respuesta Proactiva).

J. Buffer Overflow

Ataque que se basa en la ejecución de un código arbitrario en un programa al enviar un caudal de datos mayor que el que puede recibir. Un programa que admite datos de entrada con parámetros, los almacena temporalmente en una zona de la memoria denominada búfer. El problema es que algunas funciones no admiten este tipo de desbordamiento y provocan el bloqueo de la aplicación, lo que permite la ejecución del código arbitrario del atacante y permite el acceso al sistema.

Mecanismo de funcionamiento

Los pasos que rigen el comportamiento habitual de este ataque son los siguientes:

El servidor reserva unas posiciones de memoria para almacenar los datos que le llegan por la red. Para estimar el tamaño de la memoria a reservar, el programador comprueba cuál es el tamaño máximo de los datos que le pueden llegar según el estándar del protocolo implementado. En principio, se supone que no deberían llegar datos de tamaño superior al máximo permitido por el estándar.

- Un atacante genera un paquete de datos malicioso con un tamaño superior al máximo permitido.
- El servidor recibirá esos datos y los irá almacenando en la memoria reservada. Cuando el servidor llena el espacio reservado se pueden dar dos situaciones:

El servidor comprueba la situación y advierte que el tamaño de los datos enviados supera el máximo permitido, por lo que rechaza la

petición por no ser conforme al estándar. En este caso, no se da opción a la realización de un ataque.

El servidor no comprueba la situación y sigue transfiriendo los datos a memoria, sin darse cuenta de que los está enviando a posiciones de memoria que no han sido reservadas y que, por lo tanto, está sobrescribiendo posiciones de memoria. Es en este caso, cuando se puede intentar realizar el ataque.

- Si el servidor no comprueba el desbordamiento de buffer, el comportamiento más normal es intentar sobrescribir posiciones de memoria protegidas por el sistema operativo, caso en que el propio sistema operativo aborta la ejecución del servidor con un programa de error. No obstante, existe un caso en el que se sobrescriben posiciones de memoria especiales, las correspondientes a la pila del sistema operativo, que contienen las direcciones de las instrucciones que el sistema operativo va a ejecutar.
- Si el atacante consigue que uno de los datos que se desborda del buffer sobrescriba la pila del sistema operativo, puede forzar la escritura de la dirección del buffer de memoria donde el servidor ha ido almacenando los datos que le ha enviado previamente el atacante.
- Por último, el sistema operativo intentará ejecutar el código apuntado desde la pila, es decir, intentará ejecutar el contenido de los datos que el atacante ha enviado previamente. Si estos datos contienen instrucciones que permiten al atacante hacerse con el control del sistema, el sistema operativo lo ejecutará y el atacante habrá realizado con éxito un acceso remoto no autorizado al sistema.

Detección

Como técnicas de detección de este tipo de ataques se distinguen:

- El sistema ejecuta procesos o acciones no ordenadas por el administrador o el usuario del sistema.
- Modificación de información del sistema.
- Aumento del consumo de disco en momentos no esperados.

Acciones de respuesta

Entre las acciones de respuesta que puede llevar a cabo un AIRS ante este tipo de ataque se encuentran:

- Eliminar el proceso que se está ejecutando con el trozo de código malicioso (Respuesta Activa de Protección y Respuesta Activa de Recuperación).
- Devolver los activos afectados a su estado previo al ataque (Respuesta Activa de Recuperación).

K. Denial of Service

Ataque cuyo objetivo es saturar los recursos de la máquina víctima de tal forma que se inhabilitan los servicios prestados por la misma durante un periodo de tiempo indefinido. Hace inaccesible un servicio o recurso a los usuarios legítimos, provocando normalmente, la pérdida de la conectividad de la red por el consumo del ancho de banda de la red del sistema atacado o sobrecarga de los recursos computacionales del mismo.

Mecanismo de funcionamiento

El ataque se puede dar de muchas formas, pero todas utilizan el protocolo TCP/IP para llevarse a cabo. Se genera mediante la

saturación de los puertos con flujo de información, haciendo que el servidor se sobrecargue y no pueda seguir prestando servicios a los usuarios habituales, o el sistema se vuelva inestable.

Como evolución del DoS, ha surgido el ataque DDoS (Distributed Denial of Service), una ampliación del ataque DoS, que se lleva a cabo instalando varios agentes remotos en muchos ordenadores localizados en diferentes puntos (o en el mismo). El invasor consigue coordinar los agentes para incrementar de forma masiva la saturación de información, hasta el punto de conseguir lanzar un ataque de cientos de máquinas dirigido a una máquina o sistema objetivo. Esta técnica se ha revelado como una de las más eficaces y sencillas a la hora de colapsar servidores.

El objetivo de ambos ataques no reside en recuperar, modificar o destruir datos, sino que se trata de un ataque a la imagen y reputación de las compañías afectadas. Puede ser que los atacantes necesiten que un sistema caiga para que un administrador lo reinicie. Es muy fácil vulnerar un sistema justo durante el reinicio, antes de que todos los servicios estén totalmente operativos.

No se trata de ataques muy complicados pero son ataques eficaces contra cualquier tipo de equipo o sistema operativo.

Tipos de ataques DoS

Hay varias formas de expresión de un ataque de DoS (Basados en host, basados en red o distribuidos):

- **Jamming o Flooding:** ataque que consiste en desactivar o saturar los recursos de un sistema. El atacante satura el sistema con mensajes



que requieren establecer conexión pero los mensajes contienen falsas direcciones IP origen (en lugar de la dirección IP del emisor). El sistema responde a esta falsa dirección y al no recibir respuesta acumula buffers que contienen información de las conexiones abiertas. Al almacenarse muchas de estas conexiones, no hay lugar para las peticiones de los usuarios legítimos. Ejemplos: el conocido ping de la muerte o el hecho de que un atacante consuma toda la memoria o espacio en disco disponible o envíe tal cantidad de tráfico a la red que esta quede inutilizada para el resto de usuarios.

- **Syn Flood:** ataque de DoS más famoso que se basa en el modo de funcionamiento del protocolo TCP. Consiste en que el atacante envía un paquete SYN a la máquina objetivo, que le envía un ACK; el atacante no responde a dicho ACK lo que provoca que la máquina destino (víctima del ataque) espere un cierto tiempo antes de cerrar la conexión. Si se crean muchas peticiones incompletas de conexión, la máquina objetivo, normalmente un servidor, estará inactivo mucho tiempo esperando respuesta lo que produce una negación de servicio al resto de intentos de conexión. Además, muchos sistemas operativos tienen un límite muy bajo del número de conexiones "semiabiertas" que pueden manejar en un momento determinado. Si se supera ese límite, el servidor dejará de responder a las nuevas peticiones de conexión que le vayan llegando. Este ataque suele combinarse con el IP Spoofing, para ocultar el origen del ataque.

Connection Flood: ataque que se basa en explotar la limitación en el número de conexiones simultaneas que pueden sostener los ISP. Una



vez alcanzado el máximo límite permitido, el ISP no admitirá conexiones nuevas. El atacante inicia numerosas conexiones hasta que supera el límite, impidiendo al resto de usuarios legítimos establecer conexiones con el ISP víctima y dejando fuera de servicio al servidor.

- **Net Flood:** ataque que consiste en que el atacante envía tantos paquetes de solicitud de conexión que las conexiones de usuarios legítimos no pueden ni siquiera intentar tener éxito. La red atacada no puede hacer nada puesto que aunque filtre el tráfico en sus sistemas, sus líneas estarán saturadas de tráfico malicioso, inhabilitándolas para cursar tráfico útil.

- **Land Attack:** ataque que se basa en explotar un error en la implementación de la pila TCP/IP de las plataformas Windows. El ataque consiste en mandar a algún puerto abierto de un servidor (generalmente al 113 o al 139) un paquete, maliciosamente construido, con la dirección y puerto origen igual que la dirección y puerto destino. Una vez que ha pasado un número considerable de mensajes enviados-recibidos la máquina termina colgándose.

- **Supernuke o Winnuke:** ataque característico de los equipos con Windows que hace que los equipos que escuchan por el puerto UDP 137 a 139 (utilizados por los protocolos Netbios de Wins), queden fuera de servicio o disminuyan su rendimiento al enviarle paquetes UDP manipulados. Generalmente se envían fragmentos de paquetes, que la máquina víctima detecta como inválidos pasando a un estado inestable.

- **E-mail Bombing-Spamming:** ataque que consiste en enviar muchas veces un mensaje idéntico a una misma dirección, saturando así la cola de entrada de mail del destinatario. El Spamming, en cambio se refiere a enviar el e-mail a miles de usuarios, hayan estos solicitados el mensaje o no.

Detección

Como indicios para detectar este tipo de ataques se distinguen:

- Aumento del número de conexiones fallidas o semiabiertas.
- Consumo de disco no correspondiente a la actividad de un usuario legítimo.
- Caída inexplicable de un sistema conectado a la red.
- Fallo general del sistema o cuelgue de procesos por falta de CPU que está siendo utilizada por el atacante.
- Redirección del tráfico de la máquina víctima hacia un agujero negro. (Ataques a los DNS y de enrutamiento).

Acciones de respuesta

Entre las acciones de respuesta que puede llevar a cabo el AIRS ante este tipo de ataque se encuentran:

- Restringir la actividad de un usuario (Respuesta Activa de Protección y Respuesta Proactiva).
- Terminar o reiniciar el servicio comprometido (Respuesta Activa de Protección y Respuesta Activa de Recuperación).
- Terminar o reiniciar el proceso sospechoso (Respuesta Activa de Protección y Respuesta Activa de Recuperación).

- Bloquear peticiones al sistema sospechosas (Respuesta Activa de Protección y Respuesta Proactiva).
- Agregar o borrar reglas de filtrado sobre un cortafuegos (Respuesta Activa de Protección y Respuesta Proactiva).
- Bloquear puertos o direcciones IP (Respuesta Activa de Protección y Respuesta Proactiva).
- Terminar conexión TCP establecida por el atacante (Respuesta Activa de Protección).
- Rastrear la conexión del atacante para recopilar información (Respuesta Activa de Reacción).

2.3.4 Sistemas de Respuesta a Intrusiones (IRS)

Los Sistemas de Respuesta a Intrusiones (Intrusion Response Systems, IRSs) son dispositivos de seguridad responsables de inferir y ejecutar una acción de respuesta frente a intrusiones detectadas por otros dispositivos de seguridad como los IDSs. Un IRS no se limita a acciones de bloqueo sobre dispositivos en línea, sino que generaliza las capacidades básicas de respuesta de un IDS, disponiendo de un amplio catálogo de acciones de respuesta que se pueden ejecutar sobre otros componentes de seguridad. De forma general, un IRS tiene tres componentes principales: el conjunto de entradas (compuesto por un conjunto de eventos, tales como informes de intrusión procedentes de los IDSs u otras fuentes de detección, información de seguridad, etc.), un proceso de razonamiento que infiere la respuesta más adecuada en función de las entradas, y un



conjunto de acciones de respuesta que pueden ser elegidas para mitigar el ataque.

Según el tipo de respuesta proporcionado, los IRS se clasifican en tres grupos:

- A. Sistemas de notificación: grupo al que pertenecen la mayoría de los sistemas de respuesta existentes en la actualidad, que se caracterizan porque la respuesta generada al detectar una intrusión consiste exclusivamente en informes y alarmas, por lo que proporcionan únicamente información al administrador del sistema indicando que ha ocurrido un ataque y se debe responder a él.
- B. Sistemas de respuesta manual: grupo al que pertenecen los sistemas de respuesta que además de informar al administrador del sistema de la existencia de una intrusión, proporcionan un conjunto de respuestas que pueden ser útiles para contrarrestar al ataque detectado. La elección de una respuesta u otra y la ejecución de la misma es tarea del administrador.
- C. Sistemas de respuesta automática: grupo al que pertenecen los sistemas de respuesta que responden inmediatamente a la intrusión sin necesidad de intervención del administrador del sistema; es el propio sistema el que selecciona y ejecuta la respuesta más adecuada, reduciendo en gran medida el tiempo de respuesta.

Además de los cortafuegos, IDSs, IPSs o IRSs, existen otras tecnologías de defensa perimetral como pueden ser los inspectores de contenido, los sistemas de detección de sondas o los honeypots/honeynets. Cada una



de ellas tiene una función muy importante en la defensa del perímetro de una red tanto de atacantes externos como internos, y la unión de todos ellos puede proporcionar un elevado nivel de seguridad a la organización en la que se instalen.

En los últimos seis años se han propuesto varias taxonomías de sistemas de respuesta automática frente a intrusiones, de las que las más relevantes son las propuestas por (Stakhanova, Basu, & Wong, 2007) y (SHAMELI-SENDI, EZZATI-JIVAN, JABBARIFAR, & DAGENAIS, 2012). Ambas basadas en (Bishop, 2003). De acuerdo con estas taxonomías, los AIRSs se pueden clasificar en función de cuatro dimensiones.

Según el tiempo de respuesta.

En base a esta dimensión de la taxonomía los AIRSs se clasifican en reactivos o proactivos:

Reactivo: aquel que ejecuta una acción de respuesta una vez que la intrusión ha sido detectada y confirmada. Dicha confirmación puede basarse en métricas de confianza del IDS o en una coincidencia completa del vector de ataque con la firma definida en la BDD de un IDS (Stakhanova, Basu, & Wong, 2007). Por lo general un IRS reactivo intenta minimizar el daño causado por una intrusión o restaurar el estado del sistema.

Proactivo: aquel que tiene la capacidad de reaccionar frente a una intrusión antes de que está comprometa al recurso objetivo. Tiene la capacidad de prever la ocurrencia de la intrusión y actuar en consecuencia



intentando controlarlo y prevenirlo mediante la ejecución de una acción de respuesta. En la actualidad varias técnicas son empleadas para prever una intrusión, algunas se basan aproximaciones probabilísticas, análisis del comportamiento del usuario, o utilización de modelos dinámicos de Markov para predecir ataques multipaso.

En la Figura 11 se observa la relación existente entre una respuesta pasiva y activa (reactiva y proactiva) representadas sobre el marco temporal de un ataque. En el instante $t(n)-t$ el sistema se encuentra en estado normal; en $t(n)$ se detecta una intrusión y el IDS envía una alerta al AIRS; y en $t(n)+t$ se ejecuta la acción de respuesta reactiva. En función de estos tiempos sobre el marco temporal del ataque, se tienen tres intervalos (N. B. Anuar, M. Papadaki, S. Furnell, & N. Clarke, 2010):

Tiempo tr antes de la alerta de intrusión ($t(n)-t < tr < t(n)$): En este intervalo de tiempo la ejecución de una respuesta proactiva juega un rol importante previniendo un ataque sobre un sistema de la red. Acciones de bloqueo y ajustes de configuración de sistemas de seguridad pueden ser desplegados en esta fase de ejecución proactiva.

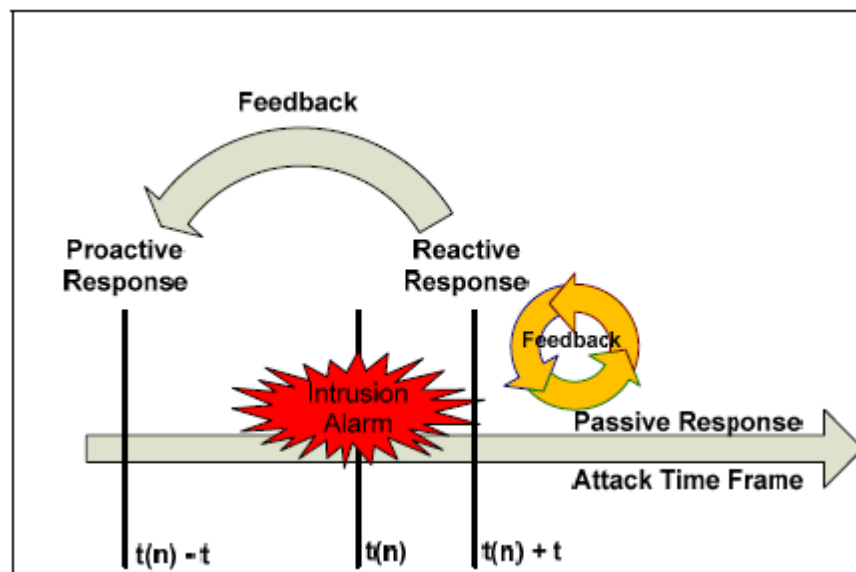
Tiempo tr después de la alerta de intrusión ($t(n) < tr < t(n)+t$): En este intervalo de tiempo se ejecuta una respuesta reactiva cuyo objetivo es minimizar el impacto causado por la intrusión. Acciones de bloqueo de tráfico, terminación de procesos, deshabilitación de usuarios, etc., pueden ser ejecutados. Para el despliegue de estas respuestas puede ser necesario interactuar con otros componentes de seguridad como:



cortafuegos, routers, aplicaciones sobre hosts, entre otros. Esta fase de ejecución reactiva termina en el instante $t(n)+t$.

Tiempo tr después de la ejecución de la respuesta reactiva ($t(n)+t < tr$): Este intervalo no tiene tiempo de finalización, ya que se trata de una fase de investigación. En esta fase el IRS evalúa y aprende de lo ocurrido en el incidente antes y después de la ejecución de la respuesta, el resultado sirve como retroalimentación para futuras respuestas.

Figura 3. Relación entre respuestas proactivas y reactivas sobre el marco temporal de un ataque.



Fuente. (N. B. Anuar, M. Papadaki, S. Furnell, & N. Clarke, 2010)

Según la capacidad de adaptación

Una vez que el AIRS ha seleccionado una respuesta, este puede evaluar o no el éxito de la misma con el fin de obtener un factor de efectividad, factor puede ser utilizado a su vez para autoajustar o adaptar el mecanismo de inferencia en posteriores ejecuciones.



En base a esta dimensión de la taxonomía los AIRSs se clasifican en estáticos o adaptativos:

Estático: aquel cuyo mecanismo de selección de respuesta permanece invariante a lo largo del tiempo, ya que no existe un seguimiento de la respuesta desplegada.

Adaptativo: aquel que tiene la capacidad de ajustarse dinámicamente. Esta capacidad de adaptación puede ser llevada a cabo de varias formas. La adaptabilidad es una característica potente de los AIRSs que permite al sistema modificar y adaptar automáticamente la elección de la respuesta en función de determinados factores, como por ejemplo, la tasa de efectividad de la respuesta en anteriores ejecuciones, o cambios producidos en el entorno.

Según mecanismo de selección de la respuesta

En base a esta dimensión de la taxonomía, un AIRS se clasifica en:

Estático: aquel en el que existe una correspondencia estática entre una intrusión y una acción de respuesta. Siempre que se detecte la intrusión de un tipo determinado el sistema inferirá la misma acción de respuesta. Estos sistemas son fáciles de construir y mantener pero son predecibles (lo que los hace vulnerables) y no tienen en cuenta otros factores importantes como el coste de la respuesta, el impacto de la intrusión, etc.

Dinámico: aquel que realiza un mapeo dinámico, es decir, asocia una alerta de intrusión a un conjunto de posibles respuestas. La selección de la mejor respuesta de entre este conjunto puede estar basada en múltiples



factores, como el estado actual del sistema, métricas de la alerta de intrusión, o una política de red.

Sensible al coste de las respuestas: aquel cuyo objetivo es seleccionar la respuesta óptima pero sin sacrificar el funcionamiento normal del sistema comprometido. Es decir, tiene en cuenta el coste y la complejidad de la respuesta además del coste del daño causado por una intrusión. En cuanto a la forma de evaluar el coste de las respuestas se proponen a su vez tres modelos, como se ve a continuación.

Según el modelo de evaluación del coste de la respuesta

En base al mecanismo utilizado para evaluar el coste de la respuesta en términos del impacto que causa su aplicación en el sistema comprometido o atacado, un AIRS se clasifica en:

Modelo estático: aquel AIRS en el que el coste de ejecutar una respuesta es obtenido mediante la asignación de un valor estático basado en la opinión de un experto.

Modelo de evaluación estática: aquel en el que el coste de ejecutar una respuesta es obtenido estadísticamente evaluando los efectos positivos y negativos de ejecutarla. Los aspectos positivos están basados en métricas de disponibilidad, confidencialidad, integridad y rendimiento de cada respuesta. Mientras que los aspectos negativos pueden basarse en la disponibilidad y rendimiento de otros recursos como consecuencia de su ejecución.

Modelo de evaluación dinámica: aquel sistema que evalúa el coste de ejecutar una respuesta de forma dinámica, en función del contexto subyacente de ese momento, como por ejemplo, el número de usuarios conectados en un momento concreto. El sistema evalúa el coste de la respuesta en función del contexto actual y cada vez que ejecuta el proceso de inferencia de una acción de respuesta.

Es importante que al evaluar el coste de ejecutar una respuesta se considere el contexto sobre el que se ejecuta, por ejemplo, tipo de ataque, importancia del host, localización del host, número de usuarios actuales en el sistema, disponibilidad de otras aplicaciones, etc.

A la taxonomía presentada por (Stakhanova, Basu, & Wong, 2007) y completada por (SHAMELI-SENDI, EZZATI-JIVAN, JABBARIFAR, & DAGENAIS, 2012), (Mateos Lanchas, 2013) añadió dos dimensiones más:

Rapidez de reacción: un sistema de respuesta a intrusiones automático debe ser lo más rápido posible, con el fin de reducir la ventana de ataque entre el momento de detección y el de respuesta. En base a esta categoría los AIRSs se clasifican entre lentos y rápidos.

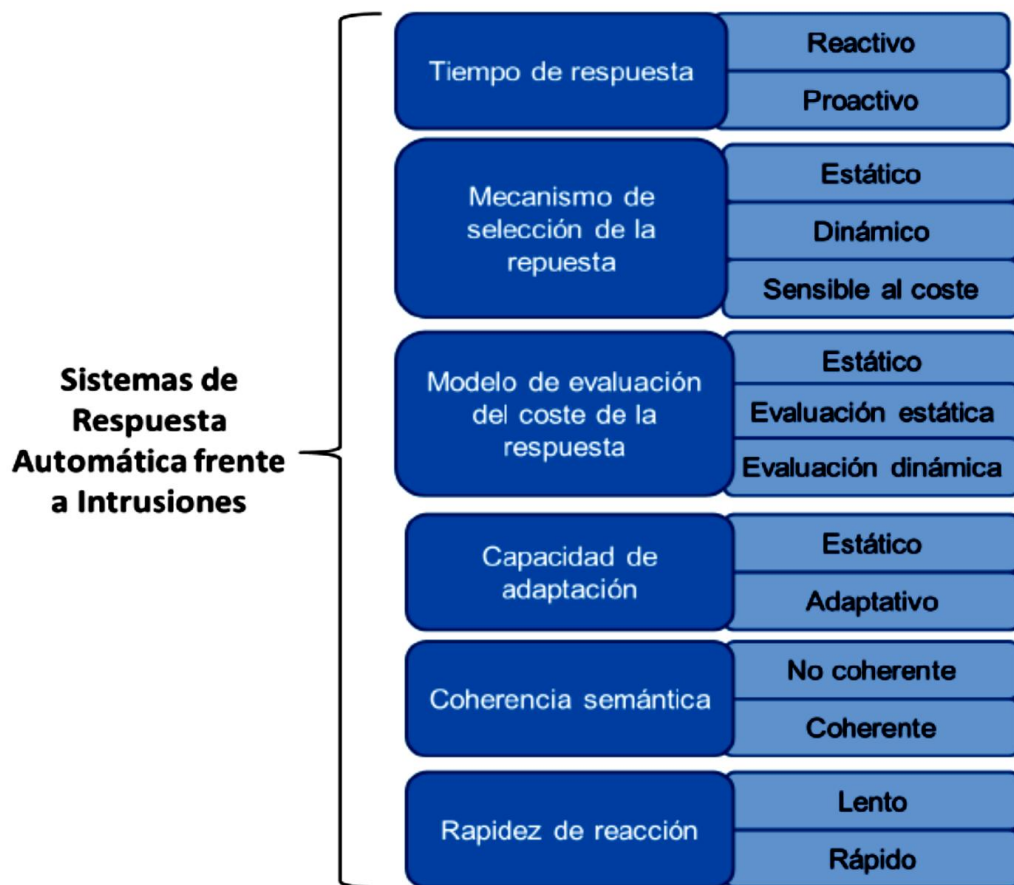
Coherencia semántica: característica muy importante en entornos de detección y reacción frente a intrusiones heterogéneos, entendiendo por coherencia semántica la capacidad del sistema de ser capaz de entender y procesar las alertas de intrusión no sólo desde un punto sintáctico sino también semántico, con independencia de la fuente de intrusión. Es decir, es la capacidad de que el sistema procese alertas de intrusión



procedentes de distintos IDSs, y que por tanto tienen formatos distintos, y sea capaz de determinar si las alertas se refieren a la misma intrusión o a diferentes.

Teniendo en cuenta las seis dimensiones en función de las cuales se puede clasificar un AIRS, lo ideal sería que este fuera proactivo, adaptativo, sensible al coste de las respuestas y con un modelo de evaluación dinámica, rápido y semánticamente coherente.

Figura 4. Taxonomía de Sistemas de Respuesta a Intrusiones.



Fuente. (Mateos Lanchas, 2013)

2.3.5 Modelos Ocultos de Markov (HMM)

Los HMMs derivan de los Procesos de Markov o Cadenas de Markov, que son un proceso estocástico. Se pueden ver como una serie de eventos, en la cual la probabilidad de que ocurra un evento depende del evento inmediato anterior. Una cadena de Markov se puede denotar por $\{X_k\}_{k \geq 0}$, donde k es un índice entero (Taylor, H. M. & Karlin, S., 1998), (Cappé, O., Moulines, E., & Rydén, T., 2005).

Los HMMs no requieren de datos anómalos para “aprender” o distinguirlos de los normales, ni requiere de reglas predictivas explícitas (Warrender, C., Forrest, S., & Pearlmutter, B., 1999). Los HMMs también son conocidos como Procesos Ocultos de Markov (HMP, Hidden Markov Processes), que son Cadenas de Markov homogéneas con estados finitos de tiempo discreto observadas a través de canales invariantes sin memoria en tiempo discreto. Donde el canal se considera formado por un conjunto finito de transacciones indexados por los estados de la cadena de Markov (Ephraim, Y. & Merhav, N., 2002).

Tipos de HMMs

En (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989) se describen a detalle los tipos de HMMs básicos, como son los Ergódicos, No Ergódicos, y los Autoregresivos. Así como también describe los modelos “izquierda-derecha”.

En el trabajo de (Nicholl, P., Amira, A., Boucha, D., & Perrot, R. H, 2008) se mencionan los diversos enfoques que los HMMs han adoptado, para ser



utilizados en el reconocimiento de rostros. Estos enfoques tratan de representar la estructura de dos dimensiones de la imagen de un rostro:

1. Uno de estos enfoques es el 1D-DHMM, de HMM Discreto de una dimensión, el cual modela la imagen de la cara usando dos HMM estándar. Uno para las observaciones en dirección vertical y otro para las observaciones en dirección horizontal.
2. Otro modelo es el Pseudo-2D HMM (2D-PHMM), el cual se utiliza para modelar la secuencia de columnas de una imagen. Cada estado que forma al modelo 2D-PHMM, es un 1D-DHMM.
3. En el mismo documento mencionan también un enfoque llamado Baja-Complejidad 2D-HMM (LC 2D-HMM), en este modelo se permiten las transiciones entre estados de manera horizontal y vertical. Se describe que la complejidad del LC 2D-HMM es más sencilla que la de 2D-PHMM. Sin embargo, su certeza en el reconocimiento es bajo.
4. Existe otro enfoque llamado HMM Jerárquico (HHMM, Hierarchical), que se utiliza en el análisis de contenido de video. Este enfoque se utiliza principalmente para modelar dominios con estructura jerárquica. El problema que presenta el algoritmo es que toma mucho tiempo de procesamiento en el análisis de las observaciones, haciéndolo impráctico para algunos dominios.
5. Un último enfoque es el HMM Estructural (SHMM), el cual maneja procesos de clustering sin supervisión, enfocado al reconocimiento de manuscritos. Este enfoque permite que las propiedades estadísticas y estructurales de un patrón, puedan ser representados en el mismo marco probabilístico.

En general, todos estos enfoques y estructuras de HMMs, mencionados en (Nicholl, P., Amira, A., Boucha, D., & Perrot, R. H, 2008) fueron empleados para el reconocimiento de rostros y texto manuscrito, así como para el análisis de contenidos de video. Por lo que los autores vieron la necesidad de modificar o ampliar la estructura de los HMMs estándar. Este trabajo propone arquitecturas orientadas al análisis de datos discretos y diseñados por los expertos del área, en base a su experiencia.

En el trabajo de (Pachter, L. et al, 2001) tratan la alineación de genes con el uso de un Par de HMM (PHMM), y para encontrar la secuencia de genes utilizan un HMM Generalizado (GHMM). En su marco de trabajo hacen una extensión de ambos modelos para generar el Par Generalizado de HMMs (GPHMM), que lo utilizan para hallar genes en cruza de especies y alineación de proteínas en el campo de la biología molecular. En esta investigación, los modelos propuestos, son diseñados en base a la experiencia. Las observaciones con las que se trabajan son secuencias de genes perfectamente determinados, es decir, son valores discretos.

HMM discretos y HMM continuos

Los procesos del mundo real producen secuencias de símbolos observables. Estos símbolos pueden ser discretos (lado de una moneda, puntos de la cara de un dado, etc.) o continuos (temperatura, velocidad de viento, ejemplos del habla, etc.). El problema consiste en modelar la señal que explique y caracterice la ocurrencia de los símbolos observados. Si tal modelo existe, entonces se puede utilizar para identificar o reconocer otras secuencias de observación. En base a la señal y la dinámica del sistema se puede decidir



la forma del modelo y el tipo de observación a estudiar: determinística o estocástica. (Rabiner, L. R. & Juang, B. H., An introduction to hidden markov models, 1986)

En base a lo anterior, existen en la literatura HMMs Discretos, caracterizados porque su función de densidad de probabilidad de observación es discreta, y los HMMs Continuos, que tienen una función de densidad de probabilidad de observación continua. En seguida, se describen las características de estos modelos.

HMM Discreto (HMMD). Este HMM tiene un alfabeto, o conjunto de símbolos discreto. Sus valores u observaciones se encuentran perfectamente definidos. Por lo que en cada estado del modelo, se encuentra definida una función de distribución de probabilidad discreta. Cada símbolo observado X tiene una distribución de probabilidad dada por la expresión $F(X) = P(X = x)$.

HMM Continuos (HMMC). En este tipo de HMM los valores a observar son de tipo continuo. Se considera que existe un espacio de observación infinito. El alfabeto no es un conjunto numerable, es decir, el conjunto de posibles valores de una observación X , abarca todo un intervalo de números reales. Además, se utilizan funciones de densidad de probabilidad (*pdf*) continuos en cada estado del modelo. Para calcular la distribución de probabilidad para una observación X continua, se utiliza la siguiente expresión $F(x) =$

$$\int_{x-e}^{x+e} F(x) dx$$

HMMs univariados y multivariados



En esta sección se presentan los conceptos de HMMs Continuos Univariados y HMMs Continuos Multivariados. También se proporciona una breve distinción entre Mezcla Gaussiana y Multidistribución Gaussiana.

A. Procesos estocásticos

El campo de la estadística tiene sus fundamentos en la teoría matemática de la ley de los grandes números y en la teoría analítica de las probabilidades. La estadística y la probabilidad tienen que ver con la incertidumbre: la probabilidad con eventos inciertos; la estadística con mecanismos inciertos. La probabilidad se hace preguntas como: “dadas estas condiciones, ¿qué tipos de salidas se pueden anticipar?”. La estadística trabaja en sentido opuesto: “¿Qué tipo de condiciones deben darse para producir estas salidas?”. Desde nuestro caso de estudio en particular, nos interesa conocer que dados ciertos eventos, o comportamientos de objetos de estudio, qué se espera o qué se puede anticipar como salida. Es decir, nos interesa estudiar el comportamiento de un conjunto de variables aleatorias a lo largo del tiempo, y sabemos que estaremos trabajando con procesos estocásticos.

En general las investigaciones presentadas en el estado del arte trabajan con procesos estocásticos (variables aleatorias), para producir modelos que nos permitan hacer predicciones sobre el comportamiento futuro de un proceso. Es decir, es una sucesión de variables aleatorias que evolucionan en función de otra variable, generalmente el tiempo.

Un ejemplo particularmente importante lo proporcionan las denominadas “Series de Tiempo” o “Series Temporales”, que registran observaciones de determinado proceso en función del tiempo. Podemos definir un proceso

estocástico como una familia $\{X(t), t \in T\}$ de variables aleatorias, clasificadas mediante un parámetro t , que varía en un conjunto T .

Ahora bien, un fenómeno físico complejo se describe con un modelo consistente en un conjunto de variables, definidas por funciones reales del tiempo. Así, llamamos determinística a una señal o magnitud que podemos predecir en cualquier instante del tiempo (dentro del intervalo de interés) y la representamos con una función real que, por lo general, es continua (en segmentos), por ejemplo: $X_d = t_0 \sin(2\pi t); t > t_0 \geq 0$

Donde t_0 es el tiempo inicial.

Por el contrario, si la señal es impredecible en cualquier instante del tiempo (dentro del intervalo de interés) la llamamos aleatoria y la representamos con una variable aleatoria que se da en función del tiempo y, por lo general, es discontinua, por ejemplo: $X_a = t_0 \sin(2\pi t + \tau); t > t_0 \geq 0$

Donde t_0 es el tiempo inicial y τ es la función de densidad de probabilidad uniforme. Entonces, podemos deducir que a un conjunto de variables aleatorias dependientes del tiempo se le llama proceso estocástico (aleatorio) y es posible representarlo por el vector del proceso estocástico $V(t) = [v_1(t), v_2(t), \dots, v_n(t)]^T$ con dimensión n .

Se puede decir que las series de tiempo univariadas han sido estudiadas ampliamente, en cambio las series de tiempo multivariadas han sido menos estudiadas. El término Multivariada o Multivariable se usa cuando existen dos o más variables siendo analizadas simultáneamente.



HMMs continuos univariados

En general la arquitectura mostrada en (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989), es el ejemplo sencillo para un HMM Discreto orientado a una serie de observación univariable. Las características son las que a continuación se describen.

El modelo consta de:

N, un número dado de estados que forman al HMM.

V, número de símbolos que componen el alfabeto.

A = a_{ij} , una matriz con una distribución de probabilidad de transición entre estados.

B = $b_{j,k}$, una matriz con una distribución de probabilidad de emisión de símbolos. Indicando la probabilidad de observar un símbolo k , en un estado j .

$\pi = \pi_i$, que es la distribución de probabilidad de iniciar algún estado i .

La única característica que marca la diferencia entre un HMM Discreto de un HMM Continuo, es la matriz **B**. Para un HMMD, existe en el modelo un conjunto finito de V símbolos o alfabeto. Con una distribución de probabilidad definida para cada símbolo o elemento observado.

En cambio, para un HMMC el alfabeto V no es un conjunto numerable. Es decir, que los valores que cada símbolo observado puede tomar, abarca un intervalo de números reales, y por lo tanto no se puede utilizar una

probabilidad de distribución similar a la de uno discreto. Por esto se utiliza una función de densidad de probabilidad (pdf) para valores continuos.

En nuestro caso de estudio se utilizó, como nuestra pdf en los HMMs Univariados, la Distribución Normal de una sola dimensión para el cálculo de la probabilidad de una secuencia de observación. La Distribución Normal, también conocida como Distribución de Gauss o Distribución Gaussiana, es la distribución de probabilidad que con más frecuencia aparece en estadística y teoría de probabilidades. Esta función tiene dos propiedades fundamentales:

Su función de densidad es simétrica y con forma de campana, lo que favorece su aplicación como modelo a gran número de variables estadísticas.

Es límite de otras distribuciones y aparece relacionada con multitud de resultados ligados a la teoría de probabilidades, gracias a sus propiedades matemáticas. Su función de densidad está dada por:

$$f_{\mu, \sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Donde μ es la Media y σ es la Desviación Estándar (σ^2 es la varianza). Cuando $\mu=0$ y $\sigma=1$, la distribución se conoce como Normal Estándar, dada por la siguiente función.

$$f_{ne}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

Muchas variables aleatorias continuas presentan una función de densidad cuya grafica tiene forma de campana. La importancia de la distribución normal se debe principalmente a que hay muchas variables asociadas a



fenómenos naturales que siguen el modelo de la normal: caracteres morfológicos de individuos, caracteres Fisiológicos, caracteres sociológicos, caracteres psicológicos, nivel de ruido en telecomunicaciones, errores en mediciones de magnitudes y valores estadísticos muestrales como la media.

En el presente trabajo de tesis se tomó como referencia la función de distribución normal por sus características mencionadas y por el teorema del límite central (que indica que en condiciones muy generales la suma de k variables aleatorias independientes, se aproxima a la función de distribución normal). Se utilizó la distribución normal en los HMMs para el cálculo de la probabilidad de los valores observados. Pero no se descarta el posible uso de otras funciones para el cálculo de probabilidad en los HMMs, incluso con la posibilidad de tener mejores resultados, de acuerdo a las series de tiempo y al análisis de los datos, que la función de distribución normal utilizada en este trabajo, para la detección de anomalías.

Para concluir, podemos decir que estos modelos de HMMs univariados únicamente toman como referencia de observación un solo símbolo, en este caso un valor real, en cada marca de tiempo t . Esto es, la serie de tiempo con la que trabajan es una secuencia sencilla de valores reales de una variable aleatoria.

HMMs continuos multivariados

Los modelos multivariados o multivariados tratan con un conjunto de valores de variables que son analizadas de manera simultánea. Los



conjuntos de datos multivariados de varias dimensiones ocurren en un gran número de dominios de problemas. Por ejemplo el comportamiento de la bolsa de valores, el patrón del habla y de la música, ciencias atmosféricas, detección de anomalías, etc. El modelado y predicción de tales datos es un problema importante en dichas áreas. Por eso es importante proponer y desarrollar, técnicas de análisis de datos confiables y flexibles para series de tiempo de datos multivariados.

Existe un número significativo de literatura acerca de series de tiempo univariados. El modelado del vector de series para dominios finitos es un área de investigación relativamente nueva. El modelado de datos mixtos (valores discretos y continuos) es todavía menos explorado debido a la complejidad de moderar interacciones entre variables finitas y valores reales

Un HMM Multivariado es un HMM que observa un vector de variables, en lugar de un solo valor (HMM Univariado). El espacio de búsqueda en un ambiente multivariado se vuelve más complejo.



B. Aplicación de los HMMs

Los Modelos Ocultos de Markov (HMMs) son procesos doblemente estocásticos con un proceso estocástico subyacente que no es observable directamente, pero que puede ser observable a través de otro conjunto de procesos estocásticos producidos por la secuencia de símbolos observados (Warrender, C., Forrest, S., & Pearlmutter, B., 1999)

La aplicación inicial y más común de los HMMs fue la del reconocimiento automático del habla, donde los HMMs fueron utilizados para caracterizar las propiedades estadísticas de una señal (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989).

Los HMMs son ampliamente usados para el reconocimiento del habla y en el modelado de secuencias de ADN. También han sido empleados en el área de bioinformática y a una gran variedad de aplicaciones fuera del área del reconocimiento del habla, tales como reconocimiento de texto manuscrito, reconocimiento de patrones en biología molecular, (Won, K.-J., Hamelryck, T., Prugel-Bennett, A., & Krogh, A, 2005) reconocimiento de rostros (Chien, J.-T. & Liao, C.-P., 2008), y reconocimiento de expresión facial (Miners, B. & Basir, O., 2005).

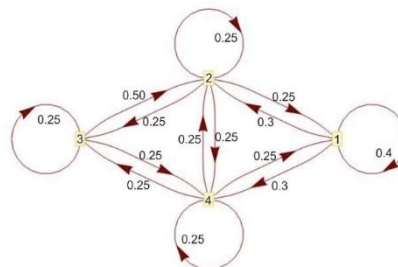
También se han utilizado en el estudio de los errores en los enlaces de red inalámbricos 802.11b, se han utilizado en proyectos de robots asistentes participando en el área de visión y navegación (López, M.-E. et al, 2005), al monitoreo de actividades humanas se han utilizado en sistemas de detección de intrusos en detección de fallas y procesamiento de imágenes.

Otros ejemplos de las diversas áreas donde se pueden emplear los HMMs son: bioinformática, visión, segmentación de música, modelado de tránsito carretero, predicción de precipitaciones, reconocimiento de rostros, secuencias de ADN, clasificación por categorías y análisis forense digital, (Pachter, L. et al, 2001), (Nicholl, P., Amira, A., Boucha, D., & Perrot, R. H, 2008).

C. Arquitectura de un HMM

Un HMM está formado por un número finito de estados conectados por transiciones. En la Figura 13 se pueden observar los estados, representado por rectángulos, que conforman al HMM. Las líneas que interconectan a los estados son las transiciones con un peso o valor de probabilidad de transitar de un estado a otro. El modelo HMM puede generar una secuencia de observaciones dependiendo de las probabilidades de su estado inicial y de sus transiciones. Esto significa que un HMM está representado por tres conjuntos de probabilidades (probabilidad del estado inicial, probabilidad de transición entre estados, y la probabilidad de observar un símbolo en cada estado).

Figura 5. Representación de un HMM con 4 Estados



Fuente. (Antolino Hernandez, 2011)



El Modelo de Markov es oculto porque no sabemos qué observación lleva a qué estado. Es decir, se desconocen los estados y las interconexiones, sólo se observan los símbolos generados por los estados.

Un HMM está definido, entre otros, básicamente por estos tres parámetros:

$A = a_{i,j}$ es la matriz de probabilidad de transición entre estados. Donde $a_{i,j} = P(q_{t+1} = S_j | q_t = S_i)$, $1 \leq i, j \leq N$.

$B = b_{j,k}$ es la matriz de probabilidad de emisión, indicando la probabilidad de observar un símbolo k , en un estado j . Es decir, $b_{j,k}$ es igual a la $P(v_k \text{ en } t | q_t = S_j)$. Para el caso continuo univariado, en este trabajo de tesis, se utiliza la siguiente ecuación.

$\pi = \pi_i$ es la distribución de probabilidad de iniciar en el estado i . Esto es, $\pi_i = P(S_i \text{ en } t = 1)$.

Por lo que un HMM puede ser definido por la tripleta (Nicholl, P., Amira, A., Boucha, D., & Perrot, R. H, 2008).

$$\lambda = (A, B, \pi)$$

En el documento de (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989) se mencionan otras características que definen a un HMM discreto:

T = Longitud de la secuencia de observación

N = Número de estados en el modelo.

M = Número de símbolos a observar.



$S = S_1, S_2, \dots, S_N$ representación de los estados.

$V = v_1, v_2, \dots, v_M$ conjunto discreto de posibles símbolos a observar.

q_t = Representación el estado actual en el tiempo t .

Estas especificaciones envuelven la selección de un HMM, como el número de estados N , los M símbolos y la especificación de λ . De todas las especificaciones π es la menos importante, y la distribución de probabilidad B es la más importante. La distribución A para algunos problemas también es importante.

Existen tres problemas básicos de interés que deben ser resueltos por los HMMs, para ser útiles en las aplicaciones del mundo real:

1. Dada la secuencia de observación $O = o_1, o_2, \dots, o_T$, y el modelo λ , ¿cómo procesamos eficientemente $P(O | \lambda)$, la probabilidad de la secuencia de observación, dado el modelo? Es decir, la probabilidad de que una secuencia de observación haya sido generada por el modelo λ .
2. Dada la secuencia de observación $O = o_1, o_2, \dots, o_T$, y el modelo λ , ¿cómo escogemos una correspondiente secuencia de estados $Q = S_1, S_2, \dots, S_T$ la cual es óptima en algún sentido (i. e. la que mejor explica las observaciones)? En otras palabras, determinar la probabilidad más alta de secuencias de estados, dada una secuencia de observación y el modelo λ .
3. ¿Cómo ajustar o reestimar los parámetros de un modelo λ para maximizar $P(O | \lambda)$?



Los HMMs tratan con estos problemas dentro de un marco probabilístico; ofrecen la ventaja de tener fuertes fundamentos estadísticos y ser computacionalmente eficientes. Pero tienen una desventaja: se necesita conocer de antemano la topología del modelo. Esto es, que para construir un HMM necesitamos decidir cuántos estados tendría y qué transiciones entre estados existirán. Una vez que la estructura del modelo está diseñada, los parámetros de transición y emisión necesitarán ser estimados con los datos de entrenamiento.

El algoritmo de Baum-Welch (BW) se usa para entrenar los parámetros del modelo. Este algoritmo utiliza un método de Maximización de Expectativa, el cual dada una configuración inicial, ajusta los parámetros del modelo a una maximización local de probabilidad con los datos. El entrenamiento con el algoritmo de BW sufre del hecho de que encuentra un máximo local, y esto es sensible a la configuración inicial de los parámetros y el rendimiento del modelo. Para más información al respecto de los HMMs, véase (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989).

Evaluación de un HMM

Dado un HMM, y una secuencia de observaciones, nos gustaría ser capaces de calcular la probabilidad de que la secuencia de observación dado un modelo es ideal. Este problema podría ser visto como una forma de evaluar la eficacia de un modelo prediciendo una secuencia de observación dado, y por lo tanto nos permitirá elegir el modelo más adecuado de un conjunto.



La probabilidad de las observaciones S para una secuencia de estado específico Q es:

$$P(O|Q, \lambda) = \prod_{t=1}^T P(o_t|q_t, \lambda) = b_{q_1}(o_1) \times b_{q_2}(o_2) \dots b_{q_T}(o_T) \quad \text{----(1)}$$

Y la probabilidad de la secuencia de estado es:

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T} \quad \text{---(2)}$$

Por lo que podemos calcular la probabilidad de las observaciones dadas el modelo como:

$$P(O|\lambda) = \sum_q P(O|Q, \lambda) P(Q|\lambda) = \sum_{q_1 \dots q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \dots a_{q_{T-1} q_T} b_{q_T}(o_T) \quad \text{---(3)}$$

Este resultado permite la evaluación de la probabilidad de O, pero para evaluar directamente sería exponencial en T.

Un mejor enfoque es reconocer que muchos cálculos redundantes serían tomadas por evaluar directamente la ecuación 3, y por lo tanto el almacenamiento en caché de cálculos puede llevar a la reducción de composición complejidad. Implementamos la caché como un enrejado de estados en cada paso de tiempo, el cálculo de la caché valorado (llamada α) para cada estado como una suma sobre todos los estados en el paso de tiempo anterior α es la probabilidad de que la secuencia de observación parcial o_1, o_2, \dots, o_t y el estado s_i en el tiempo t. Esto puede ser visualizados como en la figura 3 Definimos la variable probabilidad de forward.

$$\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = s_i | \lambda) \quad \text{---- (4)}$$



Por lo que si trabajamos a través del relleno enrejado en los valores de α la suma de la última columna del enrejado será igual a la probabilidad de la secuencia de observación. El algoritmo para este proceso se llama el **Algoritmo de Forward** y es el siguiente:

1. Inicialización:

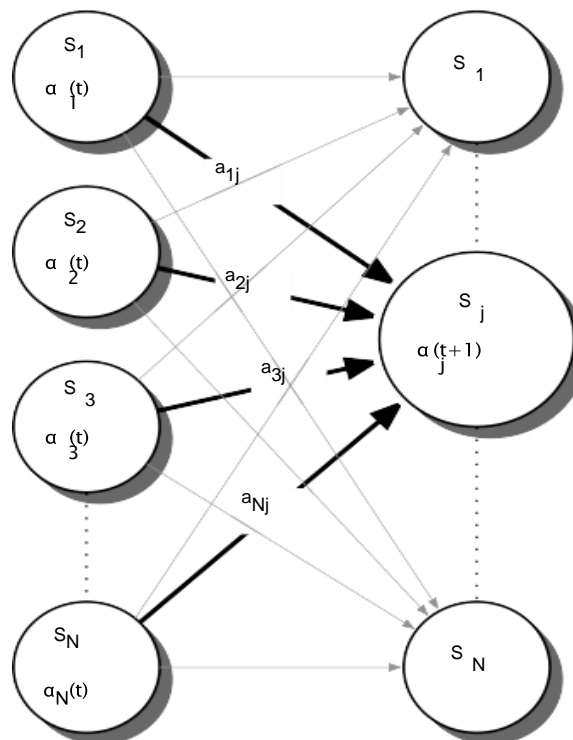
$$\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N. \text{ ---(5)}$$

2. Inducción

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(o_{t+1}), 1 \leq t \leq T - 1, 1 \leq j \leq N$$

3. Terminación

Figura 6. Etapa de Inducción del Algoritmo de Forward



Fuente. (Blunsom Phil, 2004)

Terminación de un HMM

$$P(\mathbf{O}|\lambda) = \sum_{t=1}^N \alpha_T(i)$$

La etapa de inducción es la clave para el algoritmo de forward y se representa en la figura 14. Para cada estado s_j , $A_j(t)$ almacena la probabilidad de llegar a ese estado después de haber observado la observación secuencial hasta el momento t .

Es evidente que al almacenar en caché los valores de α el algoritmo de forward reduce la complejidad de cálculos implicados a $N^2 T$ en lugar de $2NT^T$. También podemos definir un revés del algoritmo de backwards que es el inverso exacto del algoritmo de forward, hacia atrás con la variable:

$$\beta(i) = P(\mathbf{O}_{t+1} \mathbf{O}_{t+2} \dots \mathbf{O}_T | q_t = \delta_i, \lambda)$$

Como la probabilidad de la secuencia de observación parcial de $t + 1$ a T , comenzando en el estado s .

Decodificación de un HMM

El objetivo de decodificación es descubrir la secuencia de estado escondido que era más probable que tengan producido una secuencia de observación dado. Una solución a este problema es utilizar el Algoritmo de Viterbi para encontrar la mejor secuencia de estado único por una secuencia de observación. El algoritmo de Viterbi es otro algoritmo enrejado que es muy similar al algoritmo de forward, excepto que las probabilidades de transición se maximizan en cada paso, en lugar de sumados. Primero se define:



$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1 q_2 \dots q_t = s_i, o_1, o_2 \dots o_t | \lambda)$$

Como la probabilidad de la ruta de estado más probable para la secuencia de observación parcial.

El algoritmo de Viterbi es como sigue:

1. Inicialización

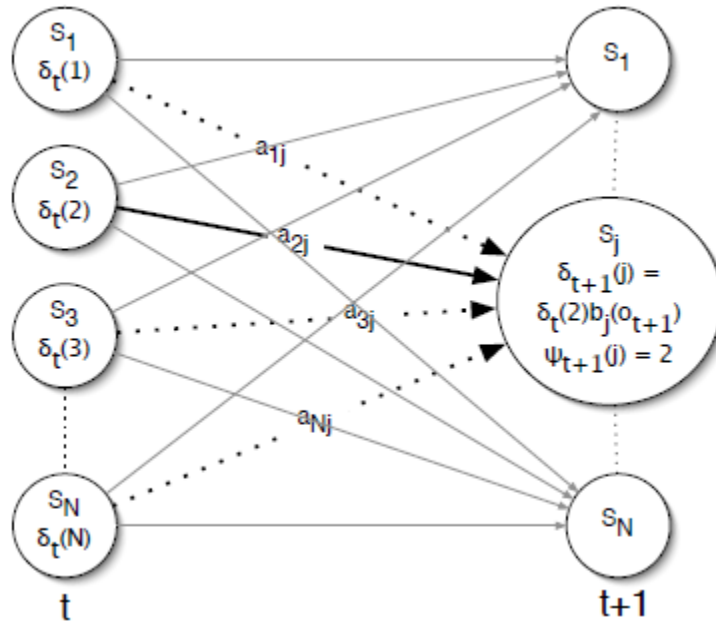
$$\delta_1(i) = \pi_1 b_1(o_1), 1 \leq i \leq N, \varphi_1(i) = 0$$

2. Recursividad

Figura 7. Recursión del Algoritmo de Viterbi

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(o_t), 2 \leq t \leq T, 1 \leq j \leq N,$$

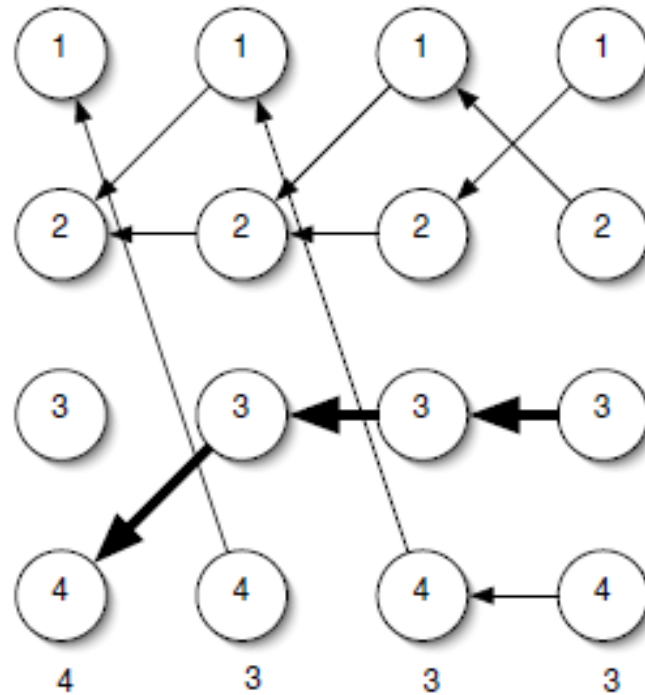
$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], 2 \leq t \leq T, 1 \leq j \leq N.$$



Fuente. (Blunsom Phil, 2004)



Figura 8. El paso Backtracing del Algoritmo de Viterbi



Fuente. (Blunsom Phil, 2004)

3. Terminación:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$

$$q^*T = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$$

4. Óptima secuencia de estado backtracking

$$q_t^* = \varphi_{t+1}(q_{t+1}^*), t = T - 1, T - 2, \dots, 1$$

El paso de recursión se ilustra en la figura 7. La diferencia principal con el algoritmo de forward es en las recurrencias paso es que estamos maximizando, en lugar de sumar, y almacenar el estado que fue elegido

como el máximo para su uso como backpointer. El paso de retroceso se muestra en la figura 8. El retroceso permite la mejor secuencia de estados que se encuentran a partir de los indicadores traseros almacenados en el paso de recursión, pero debe tenerse en cuenta que no hay manera fácil de encontrar el segundo mejor secuencia de estado.

Aprendizaje de un HMM

Dado un conjunto de ejemplos de un proceso, nos gustaría ser capaces de estimar el modelo de parametros $\lambda = (A, B, \pi)$ que mejor describen ese proceso. Hay dos enfoques estándar a esta tarea, depende de la forma de los ejemplos, que se hará referencia aquí como supervisado y sin supervisión de entrenamiento. Si los ejemplos de entrenamiento contienen tanto las entradas y salidas de un proceso, se puede realizar un entrenamiento supervisado igualando las entradas a las observaciones y resultados a los estados, pero si sólo los insumos son proporcionados en los datos de entrenamiento, entonces debemos utilizar un entrenamiento no supervisado para plantear un modelo que puede haber producido esas observaciones. En esta sección discutirá el enfoque supervisado a la formación, para una discusión sobre el algoritmo de Baum-Welch para el entrenamiento no supervisado (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989).

La solución más sencilla para crear un modelo λ es tener un gran corpus de ejemplos de entrenamiento, cada anotado con la clasificación correcta. El ejemplo clásico de este enfoque es el punto de venta tagging.

Definimos dos conjuntos:

• $t_1 \dots T_N$ es el conjunto de etiquetas, que equiparamos a la HMM estado conjunto $s_1 \dots S_N$

• $w_1 \dots W_M$ es el conjunto de palabras, que equiparamos al conjunto v observación HMM $1 \dots V_M$

así que con este modelo enmarcamos marcado parte de discurso como decodificar la más probable escondido secuencia de estado de las etiquetas de puntos de venta dada una secuencia de observación de las palabras. Para determinar el modelo parámetros λ , podemos utilizar las estimaciones de máxima verosimilitud (MLE) a partir de un conjunto que contiene frases etiquetados con sus etiquetas de puntos correctas. Para la matriz de transición que utilizamos:

$$a_{ij} = P(t_i | t_j) = \frac{\text{Count}(t_i, t_j)}{\text{Count}(t_j)}$$

Donde $\text{Count}(t_i, t_j)$ Es el número de veces t_j seguido su t_i en los datos de entrenamiento. Para la observación matriz:

$$b_j(k) = P(w_k | t_j) = \frac{\text{Count}(w_k, t_j)}{\text{Count}(t_j)}$$

Donde $\text{Count}(w_k, t_j)$ Es el número de veces w_k se ha etiquetado t_j en los datos de entrenamiento. Y, por último, la distribución de probabilidad inicial:

$$\pi_i = P(q_1 = t_i) = \frac{\text{Count}(q_1 = t_i)}{\text{Count}(q_1)}$$

En la práctica, cuando la estimación de un HMM de recuento es normalmente necesario aplicar suavizado con el fin de evitar la cuenta



cero y mejorar el rendimiento del modelo en datos que no aparecen en el conjunto de entrenamiento.

Implementación de un HMM

Cuando se implementa un HMM, de punto flotante flujo inferior es un problema significativo. Es evidente que al aplicar algoritmos de Viterbi o forward para secuencias largas con valores extremadamente pequeños de probabilidad ello acarrearía podría subutilizada en la mayoría de las máquinas. Solucionamos este problema de forma diferente para cada algoritmo:

Viterbi Underflow Como los algoritmos de Viterbi sólo multiplican probabilidades, una solución simple a subutilizada es registrar todos los valores de probabilidad y luego agregar valores en lugar de multiplicar.

De hecho, si todos los valores de las matrices de modelo (A, B, π) se almacenan conectado, luego en tiempo de ejecución sólo se necesitan operaciones de suma.

Algoritmo Forward Underflow el algoritmo de forward suma los valores de las probabilidades, por lo que es no es una solución viable para registrar los valores con el fin de evitar el desbordamiento. El más común solución a este problema es utilizar coeficientes de escala que mantienen los valores de probabilidad en el rango dinámico de la máquina, y que son sólo depende de t. El coeficiente c_t se define como:

$$c_t = \frac{1}{\sum_{i=1}^N \alpha_t(i)}$$



y por lo tanto el nuevo valor escalado para α se convierte en:

$$\hat{\alpha}_t(i) = c_t \times \alpha_t(i) = \frac{\alpha_t(i)}{\sum_{i=1}^N \alpha_t(i)}$$

Obtenido de (Phil Blunsom, 2004)

2.3.6 Algoritmos genéticos y HMMs

Los parámetros iniciales de los HMMs se determinan durante un proceso iterativo llamado “entrenamiento”. Existe un método iterativo para la optimización de los parámetros de un HMM, conocido como Algoritmo de Baum-Welch (BW). El algoritmo de BW realmente trabaja refinando o ajustando los parámetros que se han asignado al HMM. Es decir, el usuario debe diseñar la arquitectura y las conexiones entre los estados definidos, y el algoritmo de BW ajusta estos parámetros.

Otro problema que se presenta es que el algoritmo de BW puede converger en un máximo local, sin llegar a entrenar adecuadamente al modelo HMM, lo cual no es conveniente.

Una posible solución al problema de la convergencia a un máximo local, es utilizar los Algoritmos Genéticos (GAs, Genetic Algorithms). Los GAs utilizan una técnica de optimización global que puede ser usada en la optimización de los parámetros de un HMM (Korayem, M., Badr, A., & Farag, I., 2007).

Los GAs simulan el fenómeno de la evolución de la naturaleza. Su espacio de búsqueda se mapea dentro de un espacio genético. La posible solución se codifica dentro de un vector (cromosoma), y cada elemento



de este vector se le llama gen. Se calcula, en cada generación, la aptitud de cada cromosoma, los mejores cromosomas se seleccionan y, después de realizar operaciones genéticas con estos, se obtiene la solución óptima (Zhang, D. & Leckie, C, 2006). Los GAs son una técnica de optimización robusta, la cual envuelve una población de soluciones, así como un conjunto de operaciones para moverse en su espacio de búsqueda. (Korayem, M., Badr, A., & Farag, I., 2007).

La selección, cruce y mutación son tres operadores principales de los GAs, donde un individuo es el objeto de los operadores. Los GAs tienen buenas características que otros métodos clásicos no tienen por ejemplo, los métodos de Avance-Retroceso (Forward- Backward) y el de Gradiente [(Chau, C., Kwong, S., & Diu, C., et al, 1997).

2.3.7 Programación evolutiva

Es indudable que la teoría de la evolución, enunciada por Charles Darwin a mediados del siglo XIX, provocó una revolución científica. Los conceptos de esta teoría, como el de selección natural, han permeado desde entonces casi todas las disciplinas científicas. Así, puede constatarse cómo han surgido áreas de estudio que van desde la evolución química prebiótica y la evolución molecular hasta la evolución tecnológica, pasando por ideas de la evolución aplicadas a las ciencias sociales y a las ciencias cognitivas (Martínez, M., 2009).

La Computación Evolutiva es una rama de la Inteligencia Artificial que involucra problemas de optimización combinatoria; se inspira en los mecanismos de la Evolución Biológica. Durante los años 50 se

comenzaron a aplicar los principios de Charles Darwin en la resolución de problemas. Durante los años 60 y 70, varias corrientes de investigación independientes comenzaron a formar lo que ahora se conoce como computación evolutiva (Jacob, C., 2001):

Programación Evolutiva. Nació en la década de 1960 y su creador fue el Dr. Lawrence

J. Fogel, considerado el “padre de la inteligencia computacional”. Este desarrollo comenzó como un esfuerzo encaminado a crear inteligencia artificial basado en la evolución de máquinas de estado finito.

Estrategias Evolutivas. Fueron propuestas por Ingo Rechenberg y Hans-Paul Schwefel en la década de 1970. Su principal objetivo era el de optimizar parámetros.

Algoritmos Genéticos. Fueron propuestos por John Henry Holland en 1975 y su motivación inicial fue la de proponer un modelo general de proceso adaptable (Martínez, M., 2009).

La programación evolutiva toma como base la siguiente afirmación: si las poblaciones naturales se adaptan a su entorno por evolución, entonces podemos modificar selectivamente las estrategias de Solución de Problemas, codificadas como programas, y progresivamente ajustar estos programas a una tarea predefinida con un conjunto de restricciones del medio ambiente.

La evolución implica la adaptación de las poblaciones de organismos a su medio ambiente a menudo a través de métodos sofisticados de

experimentación, desarrollados en muchos casos en cientos de millones de años.

La evolución, en los sistemas computacionales, toma los principios de adaptación de la naturaleza, basado en la mutación iterada y la selección.

La evolución forma de selección iterada y variada, es el mecanismo principal en la naturaleza de adaptar individuos a un entorno específico.

Donde la adaptación es un proceso de cambio de estructura progresivo, el cual resulta en un mejor comportamiento de un individuo en su interacción con un medio ambiente (Jacob, C., 2001).

A partir de aquí se describe los algoritmos evolutivos que se pueden usar para evolucionar los parámetros de un HMM Multivariado. También se describe la forma de utilizar el HMM evolucionado en la detección de anomalías en series de tiempo o secuencia de observaciones.

Evolución de HMMs a través de Algoritmos Evolutivos

El planteamiento de creación, evolución y uso de los HMMs podrá ser usado para la detección de anomalías constituyendo un esquema completamente nuevo, a lo que existe en la actualidad en la literatura.

Existen trabajos de otros autores que han aplicado la computación evolutiva en el diseño y optimización de sistemas conexionistas. Por ejemplo (Jacob, C., 2001) presenta un esquema para evolucionar autómatas de estado finitos. (Antolino Hernandez, 2011) a través de su tesis contribuye indicando algoritmos evolutivos para evolucionar HMM los cuales se mencionas a continuación:



Para poder efectuar un proceso evolutivo, el sistema comienza creando una población de individuos, representados por cromosomas, de manera aleatoria. El número de cromosomas se define como un parámetro al momento de ejecutar el sistema.

Posteriormente, cada cromosoma HMM de la población inicial generada aleatoriamente, se tiene que evaluar con respecto a la serie de tiempo dada y la función objetivo. Con estos dos elementos se pueden calificar los HMMs, o en otras palabras, determinar su valor de aptitud.

Una vez que la población inicial ha sido evaluada, se toman los cromosomas mejor calificados y se les aplican los operadores de mutación evolutivos, como son: agregar o quitar algún estado, agregar o quitar una transición entre estados, mutar el valor de transición entre estados, mutar el valor del vector de medias y de la matriz de varianzas.

Una vez generados los cromosomas hijos, cromosomas mutados por los operadores evolutivos mencionados, se agregan a la población original, y son evaluados. Después, nuevamente se seleccionan los mejores cromosomas que pasan a la siguiente generación, donde se vuelve a repetir el ciclo del proceso evolutivo y selectivo de los mejores cromosomas de cada generación, hasta llegar al número de generación deseada.

Arquitectura de un HMM a evolucionar

Como se ha hecho mención, cada cromosoma de una población generada inicialmente, es en realidad una estructura completa de un HMM. Cada cromosoma se encuentra constituido por los siguientes genes: valor de



Aptitud, Tamaño, Matriz de Transiciones entre estados, Matriz de Medias, Matriz de Covarianzas, y el vector Pi.

La Tabla 1 muestra el cromosoma completo compuesto de los genes mencionados, y a continuación se describen las características particulares de cada gen.

Tabla 1. Arquitectura de un HMM o Cromosoma

Aptitud	Tamaño	Matriz de Transiciones	Matriz de Medias	Matriz de Covarianzas	Vector Pi
---------	--------	------------------------	------------------	-----------------------	-----------

Fuente. (Antolino Hernandez, 2011)

Aptitud. Representa el valor de aptitud del individuo, después de haber sido evaluado por la función objetivo.

Tamaño. Representa el número de estados que contiene el HMM.

Matriz de Transiciones. Contiene la matriz de probabilidad de transiciones entre los estados que constituyen al HMM.

Se muestra un esquema en la Tabla 2, donde $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$; con $1 \leq i, j \leq N$. Esto es, cada celda $a_{i,j}$ representa la probabilidad de transitar a S_j en un tiempo $t + 1$, dado que en el tiempo t se está en S_i .

Con la propiedad de $\sum_{j=1}^N a_{i,j} = 1; \forall i$.

Tabla 2. Gen que representa la Matriz de Transiciones.

Estados	S1	S2	...	SN
S1	a _{1,1}	a _{1,2}	...	a _{1,N}
S2	a _{2,1}	a _{2,2}	...	a _{2,N}
.



S	$a_{N,1}$	$a_{N,2}$...	$a_{N,N}$
---	-----------	-----------	-----	-----------

Fuente. (ANTOLINO HERNANDEZ, 2011)

Matriz de Medias. Esta matriz se encuentra constituida por los vectores de los valores de las medias de los estados. Cada media representa el valor promedio aritmético de las secuencias observadas de cada variable, con las cuales se entrenan los modelos. Es decir, esta matriz contiene la media de la distribución de la matriz B de cada estado.

La matriz B en cada estado representa la probabilidad de que se emita un símbolo en cada estado. Se puede observar este esquema en la Tabla 3.

Tabla 3. Gen que guarda la Matriz de Medias del Cromosoma.

Estado	Medias		
	X_1	...	X_d
S	$\mu_{1,1}$...	$\mu_{1,d}$
S	$\mu_{2,1}$...	$\mu_{2,d}$
.
SN	$\mu_{N,1}$...	$\mu_{N,d}$

Fuente. (Antolino Hernandez, 2011)

Matriz de Covarianzas. Los valores de la matriz son utilizados, al igual que la matriz de medias, como parámetros de entrada en la función de densidad de probabilidad (pdf). La función pdf que se utiliza es la distribución multigaussiana, que se encuentra definida en cada estado que constituyen al HMM. Las matrices de medias y covarianzas componen la matriz (B), la cual permite calcular la probabilidad de observar un símbolo en un estado determinado. Se puede observar la estructura de la matriz en la Tabla 4.



Tabla 4. Gen que mantiene la Matriz de Covarianzas para cada Estado

Variables Aleatorias	X_1	X_2	\dots	X_d
X_1	$\sigma_{1,1}^2$			
X_2		$\sigma_{2,2}^2$		
.			\ddots	
X_d				$\sigma_{d,d}^2$

Fuente. (Antolino Hernandez, 2011)

Vector Pi. Este último gen (π_i), ver Tabla 5, representa la probabilidad de que la primera observación haya sido generada en el estado S_i . Donde $1 \leq i \leq N$.

Tabla 5. Gen con el Vector Pi del Cromosoma

S_1	S_2	\dots	S_N
-------	-------	---------	-------

Fuente. (Antolino Hernandez, 2011)

Función objetivo

Para poder determinar el grado de aptitud que posee cada HMM, o cromosoma de la población, con respecto a la serie de tiempo dada, utilizamos la función objetivo descrito en la Ecuación

$$P(0 | \lambda) = \sum_{i=1}^N \alpha_T(i)$$

Donde $\alpha_T(i)$ representa a la variable de avance α (Forward) (Rabiner, L. R., A tutorial on hidden markov models and selected applications in



speech recognition, 1989), la cual calcula la probabilidad total de la secuencia observada dado un HMM. Es decir, $P(O|\lambda)$ significa la probabilidad que tiene el HMM de haber sido el modelo que generó la secuencia de observación. En otras palabras, significa que tan bueno es el modelo de HMM con respecto a la secuencia dada.

A continuación se describen los algoritmos usados, así como los parámetros que necesitan para iniciar el sistema de programación evolutiva.

A continuación se describen los algoritmos necesarios para implementar operadores evolutivos que ayudaran en la mutación de la estructura de un HMM. (ANTOLINO HERNANDEZ, 2011)

Algoritmo "Evolution"

Como se podrá observar en el **Algoritmo 1 Evolution**, se reciben como parámetros de entrada: número de cromosomas Parents o tamaño de población a crear, número de descendencia o hijos, Children, y el número de generaciones Generations, que el sistema tendrá que alcanzar con las iteraciones. No existe por el momento una función de paro, el sistema llega hasta la última generación. Una descripción más detallada de estos valores se menciona a continuación:

Parents. Se fija el número de cromosomas padres que tendrá, al inicio y durante su ejecución, en el sistema durante el proceso de iteraciones.

Children. Es la cantidad de cromosomas hijos generados en cada iteración. Estos cromosomas se generan de la población con mejor



calificación y después se integran a la población para ser calificadas, sumándose a los cromosomas padres existentes. Siempre manteniendo fijo el tamaño de la población, determinado por el valor de la variable de Parents.

Generations. Determina el número de iteraciones que realizará el sistema en la búsqueda del cromosoma óptimo. Durante el paso entre las generaciones se seleccionan los mejores cromosomas entre padres e hijos, los cuales se utilizarán en las generaciones subsecuentes.

En la línea 1 simplemente se inicia una variable (g) que lleve el conteo de las generaciones creadas en el sistema.

En la línea 2 se toman otros parámetros enviados a la función, como son: los Operadores de Mutación (MutParam, Copy, MutTrans, DelTrans, AddTrans, AddState, DelState) enviados en una lista. También contiene los siguientes valores: Min, Max, pm, sSizeCov, sSizeMean, stepSize, EvaluationFunction y un valor Epsilon. Estos valores son pasados al sistema a través de la interfaz de interacción con el usuario. En la Figura 9 se puede observar la forma de declarar y enviar los valores de los parámetros al sistema.

A continuación se describe el significado y uso de estos parámetros utilizados por el sistema.

ProbMutParam. Determina la probabilidad de mutación sobre los valores de la matriz de medias y sobre la matriz de covarianza de cada estado de los HMMs.



ProbCopy. Este valor establece la probabilidad de copiar un cromosoma tomado de la población.

ProbMutTrans. Establece la probabilidad de cambiar valores de la matriz de transiciones entre estados de un HMM.

ProbDelTrans. Fija la probabilidad de eliminar transiciones existentes entre los estados de un cromosoma.

ProbAddTrans. Establece la probabilidad de poder agregar transiciones entre estados existentes de un cromosoma.

ProbAddState. Determina la probabilidad de agregar nodos o estados a los cromosomas.

ProbDelState. Establece la probabilidad de poder eliminar nodos o estados de los cromosomas.

Esta lista de operadores evolutivos, así como su valor probable de participación, se puede observar en la Figura 9, en la variable denominada Operators.

En la Figura 9 también se observan otros parámetros como el valor de epsilon, Parents, Children, y otros valores que ya han sido mencionados en el documento.

Los siguientes parámetros y significado usados en el Algoritmo 1 Evolution, son:

MinStates. Establece el número mínimo de nodos o estados (Min), que deben contener los cromosomas.

MaxStates. Este parámetro determina el número máximo de estados (Max) que puede contener un cromosoma. En la Figura 9 también se pueden observar los valores definidos para Min y Max.

Figura 9. Definición de Parámetros en el Proceso Evolutivo

```
Epsilon = 0.7;
Parents = 300;
Children = 150;
Generations = 300;

GrafTime =
  Timing[Evolution[EPHMM[Parents, PLUS, Children, Generations,
    Operators → {{MutParameters, 0.1}, {Copy, 0.1}, {MutTransitions, 0.2},
      {DeleteTransition, 0.1}, {AddTransition, 0.2}, {AddState, 0.1},
      {DeleteState, 0.2}},
    Min → 2, Max → 16, pm → 0.2, sSizeMean → 0.8, sSizeCov → 0.8,
    stepSize → 0.5, EvaluationFunction → (ForwardHMM[#] &)]
  ];
Print["Tiempo: ", GrafTime[[1]]];
Aptitud = GrafTime[[2, 1, Table[i, {i, Length[GrafTime[[2, 1]]}], 1, 1, 1]];
ListLinePlot[Aptitud]
```

. Fuente (ANTOLINO HERNANDEZ, 2011)

Algoritmo 1 Evolution (*Parents, Children, Generations*)

- 1: $g \leftarrow 0$
- 2: $Parameters_{start} \leftarrow \{EvalFunction, Operators\}$
- 3: $MeanOperators \leftarrow Parameters_{start}$
- 4: $Pop_{initial} \leftarrow GenInitPop(Parents, minStates, maxStates)$
- 5: $Pop_{evaluated} \leftarrow EvalFunction(Pop_{initial})$
- 6: while $g < Generations$ do
- 7: $Children \leftarrow BEST(Pop_{evaluated}) + MeanOperators(Pop_{evaluated})$
- 8: $Pop_{evaluated} \leftarrow EvalFunction(Children)$
- 9: $g \leftarrow g + 1$
- 10: end while
- 11: return $BEST(Pop_{evaluated})$

pm. Este valor está asociado con el parámetro ProbMutParam, pm es el valor de probabilidad de mutar, tanto valores del vector de medias como de la matriz de covarianzas, de determinados estados de un cromosoma.

sSizeMean. Es el valor del incremento en los valores de la matriz de medias.

sSizeCov. Es el valor del incremento, step size, en los valores de la matriz de varianzas.

stepSize. Es el valor que se usará como incremento en la mutación de las transiciones entre estados de los cromosomas.

EvaluationFunction. Determina el nombre de la función encargada de evaluar a los cromosomas dentro de la población dada, y determinar su valor de aptitud. Para este caso, nuestra función de evaluación se llama ForwardHMM.

Epsilon. Se maneja este valor como intervalo para poder calcular el valor de probabilidad de los valores de la serie observada, mediante una función de densidad de probabilidad.

Estos últimos valores mencionados, también pueden observarse en la Figura 9 como parte de la interfaz del usuario.

Continuando con las instrucciones del Algoritmo 1, tenemos en la línea 3 a la variable MeanOperators que almacena los nombres de los operadores de mutación que intervendrán en la evolución de los cromosomas, así como el valor del porcentaje de participación dado a cada operador de mutación proporcionado por el usuario.



En la instrucción 4 se almacena en la variable Popinitial la población de cromosomas creados inicialmente de manera aleatoria. Esta creación de la población inicial se realiza a través de la llamada a la función GenInitPop, la cual se describirá más adelante.

En la línea 5, se evalúa toda la población inicial, con la llamada a nuestra función de evaluación de aptitud (función Forward, que se describirá en paginas siguientes), esta población calificada se almacena en la variable Popevaluated.

En la línea 7 del algoritmo, dentro del ciclo while, se generan los hijos o descendientes de la población (Children). Este vector se encontrara formado por los mejores cromosomas padres mas los cromosomas mutados por los operadores de mutación, dados al sistema.

En la línea 8, la nueva población Children, que representa a los mejores cromosomas padres y cromosomas mutados por los operadores evolutivos, se evalúa con la función de aptitud, para calificar a todos los cromosomas.

Estas dos últimas instrucciones se realizan hasta llegar al término de las iteraciones, que se encuentra marcado por el valor de Generations. Al final se regresa la última población evaluada, con el cromosoma mejor calificado al inicio de la lista.

Algoritmo GenInitPop

En el Algoritmo 2 GenInitPop se genera la población inicial; GenInitPop recibe como parámetros: Parents, Min y Max. Donde Parents especifica

el número de cromosomas a crear o tamaño de la población; los parámetros Min y Max, indica el número mínimo y máximo de estados que podrían tener los cromosomas al momento de crearse.

Algoritmo 2 GenInitPop(Parents;Min;Max)

```

1: for i = 1 to Parents do
2: Pop[i] ← InitChrome(Min;Max; timeSeries)
3: end for
4: return Pop

```

En la línea número 2, después de la instrucción for del algoritmo, se establece la llamada a la función InitChrome, que es la encargada de crear cada cromosoma. La línea 4 del código del Algoritmo 2, regresa la lista (Pop) de cromosomas creados.

Algoritmo InitChrome

El Algoritmo 3 InitChrome realiza la creación de un cromosoma. Inicia con los parámetros Min y Max, los que indican el rango de valores, entre el mínimo y máximo, que podría tomar un cromosoma al momento de generar sus estados. El parámetro timeSeries contiene el conjunto de las series de tiempo con las cuales se crearla y entrenarla cada HMM.

En la línea 1 se selecciona aleatoriamente un número entre Min y Max, que se almacena en la variable nStates, la cual se encargará de indicar el número de estados que contendrá el HMM.

En la línea 2 se hace una llamada a la función FindClusters de Matemática, que nos proporciona un conjunto o número de clusters de

acuerdo a los datos más similares. Se le pasa como para metro la serie de tiempo y el número de clusters que se desean generar; el número de cluster representa el número de estados presentes en el cromosoma. La función nos regresa una lista con los conjuntos de datos similares. Esta función se utiliza para obtener clusters de los datos similares dados en la serie de tiempo, y una vez agrupados en clusters, se determina la media (μ) y la matriz de varianzas (Σ) para cada estado.

En la línea 3, es donde prácticamente se inicia la creación del HMM. Sus valores se generan a partir de la línea 4, en esta línea se visualiza el para metro llamado qualification[unDef].

Algoritmo 3 InitChrome(Min;Max; timeSeries)

```

1: nStates ← RandomInteger(Min;Max)
2: nClusters ← FindClusters(timeSeries; nStates)
3: HMM ← [
4:   qualification[unDef];
5:   size ← nStates;
6:   transitions ← List(Transitions(nStates));
7:   means ← FindMeanCluster(nClusters);
8:   covariance ← FindCovariance(nClusters);
9:   pi ← List(1=nStates)
10: ]
11: return HMM

```

que es donde se almacenara su valor de aptitud, una vez que el cromosoma sea evaluado por la función correspondiente. Por el momento, en el instante de su creación, este valor se encuentra sin definir.

En la línea 5 se asigna al parámetro size, el número de estados que contiene el HMM.

En la línea 6 se generan de manera aleatoria las transiciones entre los (nStates) estados, con la función Transitions. Manteniendo la propiedad de que la suma total de probabilidad de las transiciones de salida de cada estado es 1, en toda la de la matriz de transiciones entre estados.

En la línea 7 se calcula la Matriz de Medias, mediante la función FindMeanCluster, la cual recibe la lista de clusters de valores en nClusters y determina la media para cada estado del cromosoma.

La línea 8 es parecida a la anterior, solo que aquí se calcula la Matriz de Varianzas con la función FindCovariance.

En la línea 9 se establece de manera proporcional la probabilidad de iniciar un estado en un momento de $t = 1$, al comienzo de la ejecución del HMM.

En la línea 10 se puede ver el cromosoma o HMM que regresa el algoritmo, después de ser creado y configurado.

Algoritmo Initialization

En las líneas del Algoritmo 4 Initialization, se describe el cálculo de la probabilidad del primer vector de datos de la serie de tiempo, dado el modelo HMM. A esto se le conoce como el método de inicialización del



algoritmo Alfa (Forward) (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989). El algoritmo retorna la lista de probabilidad de la primera observación del vector de datos, de iniciar en todos los estados del HMM, en el tiempo $t = 1$.

Algoritmo 4 Initialization(HMM;Data;Epsilon)

```

1: alpha ← {}
2: nStates ← getSizeHmm(HMM)
3: for i = 1 to nStates do
4:   probab ← FuncProbMult(getMeansState(HMM; i);
                           getCovarianceState(HMM; i);Data;Epsilon)
5:   alpha[i] ← getP iState(HMM; i) * probab
6: end for
7: return alpha

```

La instrucción más importante de este algoritmo se encuentra en las líneas 4 y 5. En la 4 se llama a la función FuncProbMult, que se encarga de calcular la probabilidad de un vector de datos, dado un estado S_i del HMM en particular.

La función toma como parámetros la matriz de medias, de la cual toma únicamente el vector de medias del estado S_i , y la matriz de covarianzas del estado S_i , el vector de datos, y el valor de Epsilon establecido.

En la línea 5 se calcula la probabilidad de que inicie un estado Si, multiplicado por la probabilidad de que el estado Si observe el vector de datos en el tiempo $t = 1$. Es decir, se determina la probabilidad que al iniciar el sistema se inicie en un estado y se observe el vector de datos en ese instante

Algoritmo Induction

En el Algoritmo 5 Induction, que se utiliza a continuación del algoritmo Initialization (ver Algoritmo 4), se calcula la probabilidad de los vectores de datos restantes de la serie de tiempo. Este algoritmo calcula la probabilidad de cada uno de los datos, a partir del segundo dato en la serie, contra un modelo HMM. Este proceso, denominado inducción, necesita el vector de probabilidades alpha, generado con el algoritmo Initialization que proceso el primer dato de la secuencia, para poder calcular las probabilidades de los datos restantes en la secuencia de observación.

Al Algoritmo 5 se le conoce como el método de inducción del algoritmo Alfa (Forward). El algoritmo regresa la lista o vector de probabilidad de la serie de tiempo, verificada contra todos los estados del HMM en un tiempo de $t + 1$.

El algoritmo necesita como parámetros de entrada: el HMM, el vector de datos (Data), el vector de probabilidad inicial (alpha), y el valor del Epsilon.

En la línea 3 del algoritmo, se toman todas las transiciones existentes entre los estados del modelo, y se asignan a la variable matrix $A_{i,j}$.



En la línea 5 se calcula la probabilidad del vector de datos, de ser observado en un estado S_j en un tiempo $t + 1$.

Algoritmo 5 Induction(HMM;Data; alpha;Epsilon)

```

1: ListAlpha ← { }
2: nStates ← getSizeHmm(HMM)
3: matrixAi;j ← getAllTransitions(HMM)
4: for j = 1 to nStates do
5:     probab ← FuncProbMult(getMeansState(HMM; j);
                           getCovarianceState(HMM; j);Data;Epsilon)
6:     sumAlpha ← alpha[j] * matrixAi;j [All; j]
7: ListAlpha[j] ← sumAlfa * probab
8: end for
9: return ListAlpha

```

En la línea número 6 se calcula la probabilidad de haber iniciado en el estado S_i en el tiempo t (representado por α), por la probabilidad de transitar del estado S_i al estado S_j en el tiempo $t + 1$ (almacenado en la variable $\text{matrix } A_{i;j}$).

En la línea 7 se almacena en la variable ListAlpha la probabilidad total de haber partido del estado S_i , en un tiempo t , de haber transitado del estado S_i al estado S_j , y de observar el vector de datos en el estado S_j , en un tiempo $t + 1$.



La línea 9 muestra el regreso de la lista o vector ListAlpha con la probabilidad del dato $t + 1$, siguiente dato en la serie de tiempo, dado por el HMM.

Algoritmo Forward

El Algoritmo 6 Forward calcula la suma de la probabilidad total de la serie contra el HMM en cuestión. Este es el algoritmo que empleamos como Función Objetivo, con el cual evaluamos la aptitud de cada HMM contra la serie de tiempo dada o lista de vector de datos.

Después de evaluados todos los HMMs se ordenan y seleccionan los HMMs mejor calificados.

El algoritmo retorna el valor de probabilidad total de la serie de tiempo generada con el HMM, en la marca de tiempo $t = T$, que representa el último dato de la serie.

Algoritmo 6 Forward(HMM; timeSeries;Epsilon)

```

1: alpha ← {}
2: len ← Length(timeSeries) - 1
3: alpha[1] ← Initialization(HMM; timeSeries[1];Epsilon)
4: for t = 1 to len do
5:     alpha[t + 1] ← Induction(HMM; timeSeries[t + 1];
                             alpha[t];Epsilon)
6: end for
7: return ProbabTotal[alpha[len + 1]]

```



En la línea 3 del algoritmo se inicia el proceso de cálculo de la probabilidad, con la función Initialization, tomando el modelo HMM, el primer vector de datos de la serie de tiempo y el valor de ϵ .

En la instrucción, de la línea 5, se envían los vectores de datos subsecuentes a la función Induction, para el cálculo de las probabilidades restantes hasta $t = T$, donde T es la longitud total de nuestra serie de tiempo o longitud de la secuencia de datos dada.

Como α es una matriz, la suma de la probabilidad total del último vector contendrá la probabilidad total de la serie de tiempo, generada con el HMM. Por eso se retorna ese valor que constituye la calificación final de aptitud al HMM.

Algoritmo FuncProbMult

En el Algoritmo 7 FuncProbMult se procesan los parámetros recibidos: el vector de medias $vMean$, la matriz de varianzas $matCovar$, el vector de datos (Data) con los valores de las variables, y el valor de ϵ .

El algoritmo se encarga de calcular y retornar la probabilidad del vector de datos, utilizando para esto la función que integra el resultado de la función de densidad de probabilidad, que en este caso es la función de distribución normal multivariable. La función de distribución normal multivariable utiliza el vector de medias y la matriz de varianzas del estado S_i , y por último, el valor de ϵ . Es decir, que la función calcula la probabilidad de observar el vector de datos en el estado S_i en una marca de tiempo t . El valor de ϵ determina el intervalo a considerar para determinar la probabilidad del vector de datos dado, en el estado S_i .



Algoritmo 7 FuncProbMult (vMean; matCovar; Data; ε)

1: probab ←

$$\int_{X^*_D-E}^{X^*_D+E} \dots \int_{X^*_2-E}^{X^*_2+E} \int_{X^*_1-E}^{X^*_1+E} f(\mu, \Sigma, 0) dx_1 dx_2, \dots, dx_D$$

2: return probab

Se conoce que la probabilidad asociada a un valor específico para una variable aleatoria continua x_0 , es cero. Es por esto, que se necesita especificar un intervalo ($x_0 - \epsilon$, $x_0 + \epsilon$) para poder calcular la probabilidad de que la variable aleatoria continua pueda caer en ese intervalo.

Ahora bien, si se tiene una observación D-dimensional de las variables

$$X = \{X_1; X_2; \dots; X_D\}$$

y se desea calcular la probabilidad de que X está en la cercanía de

$$\{X^*_1, X^*_2, \dots, x^*_d\}$$

se necesita integrar D veces como se muestra en la ecuación:

$$\int_{X^*_D-Ep}^{X^*_D+Ep} \dots \int_{X^*_2-Ep}^{X^*_2+Ep} \int_{X^*_1-Ep}^{X^*_1+Ep} f(\mu, \Sigma, 0) dx_1 dx_2, \dots, dx_D$$

donde $f(\mu; \Sigma; 0)$ es la función de multidistribucion gaussiana.

Algoritmo AddTransition

El Algoritmo 8 AddTransition, es el operador de mutación que se encarga de agregar una nueva transición a un estado. Recibe los siguiente valores como parámetros de entrada: el HMM a mutar, el número de estado



nState a modificar seleccionado de manera aleatoria, y el número de espacios o transiciones availabTrans, donde puede insertarse un nuevo valor de transición.

En la línea 1 del algoritmo se obtiene la lista de las transiciones existentes del estado.

En la línea 2, se determina el valor del incremento Increm para operar con las transiciones existentes, considerando únicamente las transiciones que tienen un valor distinto de cero, más el incremento en una unidad, que corresponde a la nueva transición.

En la línea 4 determinamos de manera aleatoria, en caso de tener mas de un espacio disponible, el número de espacio o estado donde se va a insertar la nueva transición.

Algoritmo 8 AddTransition(HMM; nState; availabT rans)

```

1: transState ← getT ransState(HMM; nState)
2: Increm ← 1=(transState 6= 0) + 1
3: if availabT rans > 1 then
4: availabT rans ← RandomInteger(Length(availabT rans))
5: end if
6: for i = 1 to Length(transState) do
7: if transState[i] 6= 0 then
8: transState[i] ← transState[i] - (transState[i] * Increm/(1 - Increm))
9: end if
10: end for
    
```



11: transState[availabTrans] ← Incrém + (Incrém *Incrém / (1 - Incrém))

12: return transState

En la línea 8, dentro del ciclo for, se realiza una operación aritmética para determinar el nuevo valor que contendrán las transiciones del estado, tomando en cuenta su valor actual mas el valor del incremento. La forma de mantener de manera correcta la distribución de la unidad entre las transiciones es de la manera siguiente: al número de transiciones que tiene actualmente el estado, se le agrega una mas, y se divide la unidad entre este número ($inc = [1/(numTrans + 1)]$). inc es el incremento que se agregara a las transiciones actuales de la forma $Trans_i = Trans_i - [Trans_i * inc/(1 - inc)]$; con $i < N$ porque existen transiciones con valor de 0 en el estado.

En la línea 11 se inserta el nuevo valor de la transición, dependiendo del valor proporcional que le corresponda generado de la operación aritmética $Trans_{new} = inc + [inc * inc/(1 - inc)]$.

Y por último, se retorna la lista de transiciones actualizada para ese estado. Que de manera posterior se incorporará al HMM mutado.

Algoritmo DeleteTransition

En el Algoritmo 9 DeleteTransition, describe al operador de mutación que elimina una transición de un estado. Este algoritmo recibe como único parámetro de entrada el HMM sobre el cual va a operar para mutar.

En la línea 1 se selecciona aleatoriamente un numero de estado al cual se le eliminara la transición, el número de estado se almacena en la variable delTrans.



En la instrucción 2 del algoritmo se guarda la matriz de transiciones del HMM en la variable allTrans.

En la siguiente instrucción se elige aleatoriamente el número preciso de la transición a eliminar delete, del estado seleccionado previamente.

En la línea 4 se actualiza con un cero, lo que significa que se elimina la transición, existente en el estado seleccionado.

En la instrucción 5 se normalizan los valores restantes del vector de transiciones, de tal forma que sumen la unidad. Esto es, de manera aleatoria se distribuyen nuevos valores a las transiciones activas, manteniendo el valor de 1 en la distribución.

En la línea 6 se actualiza la matriz de transiciones del modelo HMM, y finalmente se retorna el HMM con los valores mutados.

Algoritmo MutParameters

El Algoritmo 10 MutParameters, se encarga de operar sobre los valores del vector de medias y los de la matriz de covarianzas. Incrementando o decrementado los valores guardados en el vector de medias y en la matriz de covarianzas de manera aleatoria, de acuerdo a un valor de regresado por una función de distribución normal. El algoritmo recibe como único parámetro el HMM a mutar.



Algoritmo 9 DeleteTransition(HMM)

```

1: delTrans ← RandomInteger(getSizeHmm(HMM))
2: allTrans ← getAllTrans(HMM)
3: delete ← RandomInteger(getSizeHmm(HMM))
4: allTrans(delTrans; delete) ← 0
5: allTrans[delTrans] ← NormalizeTransition(allTrans(delTrans))
6: HMM ← allTrans
7: return HMM

```

Como se podrá observar dentro del primer ciclo for, en las líneas 2 y 3 se obtienen el vector de medias y la matriz de varianzas del estado en cuestión del HMM.

En la instrucción de la línea 4 se genera un valor de tipo real con la función RandomReal, el cual se compara contra el parámetro pm (probabilidad de mutación), establecido y descrito previamente. La comparación determina si se procede o no a modificar los valores del vector de medias y la matriz de covarianzas del estado.

Algoritmo 10 MutParameters(HMM)

```

1: for j = 1 to getSizeHmm(HMM) do
2: vMean ← HMM
3: matCovar ← HMM
4: if RandomReal() < pm then
5: for k = 1 to numVars do

```

```

6: vMean[j; k] ← vMean[j; k] + NormalDistribution(sSizeMean)
7: matCovar[k; k] ← matCovar[k; k] + NormalDistribution(sSizeCov)
8: end for
9: end if
10: end for
11: HMM ← vMean
12: HMM ← matCovar
13: return HMM

```

En las líneas 6 y 7, se procede a modificar los valores del vector de medias y la matriz de varianzas, al sumarle los valores generados de manera aleatoria por una distribución normal. A la función de distribución normal se le pasan los parámetros `sSizeMean` y `sSizeCov`, como se pueden ver en las líneas 6 y 7. Los valores de `sSizeMean` y `sSizeCov` fueron establecidos por el usuario al iniciar el sistema evolutivo y representan pequeños desplazamientos del valor de las medias y de la matriz de varianzas. Los incrementos en los valores de desplazamiento se realizan de acuerdo a los fijados previamente como parámetros de inicio del sistema.

Una vez realizada esta operación iterativa para cada estado del HMM, en las instrucciones de las líneas 11 y 12 se transfiere el vector de medias y matriz de varianza al HMM. En la línea 13, el algoritmo regresa el HMM con sus valores mutados.

Algoritmo MutTransitions

El Algoritmo 11 MutTransitions, se encarga de llevar a cabo la mutación de los valores de las transiciones de un estado en particular. La forma de operar es como sigue: en la línea 1 se determina el número de estados que tiene el HMM.

Algoritmo 11 MutTransitions(HMM)

```

1: nStates ← getSizeHmm(HMM)
2: mutState ← RandomInteger(nStates)
3: transState ← getTransState(HMM; mutState)
4: for i = 1 to nStates do
5: transState[i] ← transState[i] + NormalDistribution(stepSize)
6: end for
7: HMM ← NormalizeTransition(transState)
8: return HMM

```

En la línea 2 se selecciona de manera aleatoria el número de estado sobre el cual operar (mutState). En la siguiente instrucción se obtienen las transiciones de ese estado, y se almacenan en el vector transState.

En la línea 5, dentro del ciclo iterativo for, se modifican todos los valores de las transiciones que tiene el estado, almacenado en el vector transState. El valor que se asigna es el que se crea de manera aleatoria con una función de distribución normal (NormalDistribution).

La variable `stepSize` es un parámetro definido y descrito de manera previa, que es un valor declarado por el usuario al inicio del programa como parámetro.

En la línea 7, una vez modificados los valores de las transiciones del estado, se normalizan los valores, de tal manera que la suma del vector de transiciones sea igual a la unidad. En la última línea se regresa el HMM modificado o evolucionado aleatoriamente.

Algoritmo DeleteState

El Algoritmo 12 DeleteState realiza la operación de mutación que se encarga de eliminar un estado de un HMM dado.

Algoritmo 12 DeleteState(HMM)

```

1: nStates ← getSizeHmm(HMM)
2: randState ← RandomInteger(nStates)
3: nStates ← nStates - 1
4: allTrans ← Delete(getAllTrans(HMM); randState)
5: for i = 1 to nStates do
6: allTrans[i] ← Delete(allTrans[i]; randState)
7: end for
8: HMM ← NormalizeTransition(allTrans)
9: HMM ← nStates
10: nClusters ← FindCluster(timeSeries; nStates)
11: HMM ← FindMeanCluster(nClusters)
12: HMM ← FindCovariance(nClusters)
    
```



13: HMM ← NormalizeT ransition(Delete(getPiHmm(HMM); randState))

14: return HMM

En las líneas 1 y 2, se obtiene el número de estados con que cuenta el HMM y se selecciona aleatoriamente un número de estado a eliminar.

En la instrucción número 4 se procede a eliminar la lista o vector de transiciones del estado seleccionado (randState), de la matriz de transiciones del modelo HMM representado por getAllTrans(HMM). Y con esta instrucción se elimina el vector de transiciones del estado de la matriz.

En la instrucción 6, dentro del ciclo iterativo del for, se procede a eliminar toda transición existente hacia el estado eliminado, de todos los vectores de los estados restantes que aun se encuentran en la matriz de transiciones.

En la línea 8 se procede a normalizar el valor de todas las transiciones de los estados restantes. Es decir, que la suma de los valores de cada vector de transiciones sea igual a 1.

En la línea 9 se actualiza el número de estados que ahora tiene el HMM. Y en la línea 10 se determina el número de clusters en base a la serie de tiempo y el nuevo número de estados del HMM.

En las líneas 11 y 12 del algoritmo se determina el nuevo vector de medias y las nuevas matrices de varianza para cada estado del HMM.

En la línea 13 se procede a eliminar el valor de P_i del estado borrado, y se normalizan los valores de P_i de los estados restantes. Por último se regresa el HMM modificado.

Algoritmo AddState

En el Algoritmo 13 AddState, se realiza la operación de mutación que consiste en agregar un nuevo estado a un cromosoma o HMM.

Algoritmo 13 AddState(HMM)

```

1: nStates ← getSizeHmm(HMM) + 1
2: nClusters ← FindClusters(timeSeries; nStates)
3: HMM ← nStates
4: HMM ← newTransState(getAllTrans(HMM))
5: HMM ← FindMeanCluster(nClusters)
6: HMM ← FindCovariance(nClusters)
7: HMM ← Pi(1=nStates)
8: return HMM

```

En la línea 1 se obtiene el número de estados actual, y se incrementa el valor en uno. Lo cual indica que el HMM aumentara su tamaño en un estado adicional.

Con esta información, se determina el número de clusters en la línea 2. Tomando como datos principales la serie de tiempo (timeSeries) y el nuevo número de estados (nStates).

En la instrucción 3, se actualiza el nuevo número de estados del HMM. Y en la línea 4 se realiza la creación del vector de transiciones, que



contendría el nuevo estado creado, hacia los estados existentes. Estas conexiones son creadas de manera aleatoria, y sus valores son normalizados. En los estados existentes, no se establece la conexión hacia el nuevo estado de manera inmediata, sino que se deja que el mismo sistema siga evolucionando, y que en una generación futura, sea probable que se se aplique el operador de agregar una nueva transición a ese estado.

En las líneas 5 y 6 se vuelve a determinar el vector de medias y la matriz de varianzas para todos los estados, debido a la inserción del nuevo estado.

En la línea 7 se recalcula la probabilidad de iniciar algún estado en un tiempo de $t = 1$. Es decir, la probabilidad de que el sistema tome al iniciar, cualquier estado como punto de arranque. Y por último se regresa el HMM mutado.

2.4 Definición de terminología

Proactividad: Capacidad de reaccionar frente a una intrusión antes de que está comprometa al recurso objetivo. Deseable que tenga la capacidad de prever la ocurrencia de la intrusión y actuar en consecuencia intentando controlarlo y prevenirlo mediante la ejecución de una acción de respuesta (Bishop, 2003)

Algoritmo: Es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.

Ontología: Las ontologías son una técnica utilizada en informática en el mundo de la gestión del conocimiento e inteligencia artificial como modo de representar formalmente conceptos, su significado y relaciones entre ellos.

Técnica: es un procedimiento o conjunto de reglas, normas o protocolos que tiene como objetivo obtener un resultado determinado.

Métrica: Forma de medir y una escala, definidas para realizar mediciones de uno o varios atributos.

Sistemas de Detección de Intrusiones: Sistemas que se encargan de monitorizar eventos en busca de signos de comportamiento malicioso o inesperado. Su función es detectar accesos no autorizados a una red informática.

Sistema de Prevención de Intrusiones: Sistemas que además de analizar la red en busca de actividad maliciosa y registrar dicha actividad, intentan bloquear o detener la intrusión con acciones de respuesta de protección

básicas (terminación de conexiones, procesos y accesos asociados a la intrusión, restablecimiento de conexión, bloqueo de tráfico de la IP atacante, etc.).

Sistema de Respuesta Automática frente a Intrusiones (AIRS): Sistemas de respuesta que responden inmediatamente a una intrusión sin necesidad de intervención del administrador del sistema; es el propio sistema el que selecciona y ejecuta la respuesta más adecuada, reduciendo en gran medida el tiempo de respuesta.

Predicción de Intrusiones: Modelo capaz de distinguir entre las conexiones malignas, llamadas intrusiones o ataques, y conexiones normales y cuyo objetivo es ejecutar una acción de respuesta frente a una intrusión antes de que esta se lleve a cabo.

CAPITULO III: MARCO METODOLÓGICO

3.1 Tipo y diseño de la investigación

La presente investigación es del tipo **Tecnológica aplicada** y su diseño **Cuasi-Experimental**.

3.2 Población y muestra

Población

La población estará conformada por el consumo de ancho de banda de la red del centro de datos de la Municipalidad Distrital de Chongoyape

Muestra

La muestra estará conformada por una serie de tiempo de 168 valores, que corresponden al consumo de ancho de banda en Kbit/segundo monitoreada cada hora en 7 días consecutivos. Los datos de la semana comprenden del lunes 06 de junio al domingo 12 de junio de 2016.

3.3 Hipótesis

La aplicación de los Modelos Ocultos de Markov y Programación Evolutiva dotará de Proactividad a un Sistema de Respuesta Automático a Intrusiones en redes informáticas.

3.4 Operacionalización

Variable Dependiente
Modelos Ocultos de Markov
Programación Evolutiva

Variable Dependiente	Dimensiones	Indicadores	Técnicas e
			Instrumentos de Recolección de Datos
Proactividad	Capacidad de Reacción	Tiempo de despliegue de una acción de respuesta	Observación Ficha de Registro de Eventos
		Eficiencia de la Reacción	Análisis Documental Ficha de Registro de Datos
	Capacidad de Prever	Fiabilidad de detección de ataques	Grado de fiabilidad con el cual, la detección de ataques es acertada



3.5 Métodos, técnicas e instrumentos de recolección de datos

Para la recolección de datos se usará las siguientes técnicas.

Análisis

Mediante esta técnica, se pretende analizar los diferentes ataques en las redes informáticas así como actúan frente a ellos los AIRS. Esto será de ayuda para seleccionar las muestras más idóneas para el entrenamiento del sistema y obtener mejores resultados.

Observación

La observación es la técnica de investigación básica, sobre las que se sustentan todas las demás, ya que establece la relación básica entre el sujeto que observa y el objeto que es observado, que es el inicio de toda comprensión de la realidad.

Entrenamiento y prueba del sistema

Las técnicas evaluadas serán evolucionadas a través de la Programación Evolutiva para obtener el comportamiento proactivo, dicha evolución será almacenada en una base de datos del AIRS a partir de la cual se ira entrenando para poder responder y predecir satisfactoriamente intrusiones en redes informáticas.

3.6 Procedimiento para la recolección de datos

Diariamente se inspeccionara visualmente los informes de intrusión en formato syslog, emitidos por los IDS implementados en la plataforma virtual donde se montara la arquitectura propuesta, se insertara intencionalmente ataques a la aplicación para verificar el rendimiento y validación de la propuesta, las acciones tomadas por el AIRS Proactivo propuesto serán guardadas en una base de datos para posteriormente analizar y determinar el rendimiento de la aplicación construida.

3.7 Plan de análisis estadístico de datos

Para la evaluación de cada algoritmos utilizado en cada prueba, se usara la Observación Directa para luego hacer el cálculo usando las siguientes formulas estadísticas.

Desviación Estándar

La desviación estándar o desviación típica (σ) es una medida de centralización o dispersión para variables de razón (ratio o cociente) y de intervalo, de gran utilidad en la estadística descriptiva.

Se define como la raíz cuadrada de la varianza. Junto con este valor, la desviación típica es una medida (cuadrática) que informa de la media de distancias que tienen los datos respecto de su media aritmética, expresada en las mismas unidades que la variable. Su fórmula es:

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{N}}$$



Media

Es el valor que ocupa el lugar central de todos los datos cuando éstos están ordenados de menor a mayor.

La mediana se representa por M_e y solo se puede hallar sólo para variables cuantitativas.

$$M_e = L_i + \frac{\frac{N}{2} - F_{i-1}}{f_i} \cdot a_i$$

Moda

En estadística, la moda es el valor con una mayor frecuencia en una distribución de datos.

$$M_o = L_i + \frac{f_i - f_{i-1}}{(f_i - f_{i-1}) + (f_i - f_{i+1})} \cdot a_i$$

3.8 Criterios éticos

Tabla 6. Criterios Éticos.

Criterios	Características éticas del criterio
Confidencialidad	Se asegura la protección de la identidad de las personas u organizaciones que lo solicitan.
Objetividad	El desarrollo de la plataforma propuesta se basara en criterios técnicos e imparciales
Originalidad	Se citaran las fuentes bibliográficas de la información mostrada, a fin de demostrar inexistencia de plagio intelectual.
Veracidad	La información mostrada será verdadera, cuidando la confidencialidad de esta.
Consentimiento informado	El participante está de acuerdo con ser informante y conocer sus derechos y responsabilidades.

Fuente: Elaboración Propia

3.9 Criterios de rigor científico

Tabla 7. Criterios de rigor científico.

Criterios	Características científicas del criterio
Validación	Se validara los instrumentos de recolección de datos y la arquitectura propuesta.
Confiabilidad	Se realizaran los cálculos estadísticos para la determinación del nivel de consistencia interna de los instrumentos de recolección de datos.
Contrastación	Se contrastará la hipótesis a través de métodos estadísticos.

Fuente: Elaboración Propia



CAPÍTULO IV: ANALISIS E INTERPRETACION DE RESULTADOS

4.1 Resultados en tablas y gráficos

En esta sección se demuestra que la evolución de HMMs es un proceso de optimización de parámetros de los modelos, que no requiere de conocimiento previo para lograr encontrar un modelo óptimo. En cambio en la optimización de los parámetros de un HMM con el algoritmo de Baum-Welch, depende en gran medida de sus parámetros iniciales. Es decir, requiere de conocimiento previo del análisis de los datos y la estructura a diseñar de los modelos.

La Tabla 8 describe la detección de comportamiento anómalo presente utilizando HMM-BW. Esta tabla presenta el porcentaje de Aciertos, Falsas Alarmas o Falsos Positivos, y las anomalías no reconocidas (Falsos Negativos)

Tabla 8. Resultado de la detección usando HMM basado en Baum-Welch

Tamaño de Ventana	%		
	Aciertos	Falso Positivo	Falso Negativo
3	57	43	0
4	60	40	0
5	59	41	0
6	65	35	0

Fuente: Elaboración Propia



En los datos de la Tabla 8 se puede observar que la ventana de datos que presento mejor desempeño en la detección, fue la ventana de 6 datos con un 65% de aciertos en la detección de anomalías, 35% de falsos positivos y cero por ciento de falsos negativos. La peor ventana de datos fue la de tamaño de 3 datos, con 57% de aciertos en la detección de anomalías, 43% de falsos positivos y cero por ciento de falsos negativos.

Con los mismos datos se realizó el experimento con HMM basado en EP, la Tabla 9 presenta el porcentaje de Aciertos, Falsas Alarmas o Falsos Positivos, y las anomalías no reconocidas (Falsos Negativos).

Tabla 9. Detección de anomalías usando HMMCU basado en EP

Tamaño de Ventana	%		
	Aciertos	Falso Positivo	Falso Negativo
3	89	7	4
4	93	2	5
5	89	4	7
6	84	5	11

Fuente. Elaboración Propia

El experimento reporta el mejor resultado con un tamaño de ventana de 4 datos, usando un HMM basado en EP, con un 93% de detecciones, 2% de falsos positivos y un 5% de falsos negativos. El peor resultado fue producido por un tamaño de ventana de 6 datos, con 84% de aciertos en la detección de anomalías, 5% de falsos positivos, y un 11% de falsos negativos.

En resumen se puede concluir que los HMMs generados con programación evolutiva tuvieron mucha mayor eficiencia que los entrenados con el Algoritmo Baum-Welch. El algoritmo de Baum-Welch solo refina u optimiza los valores

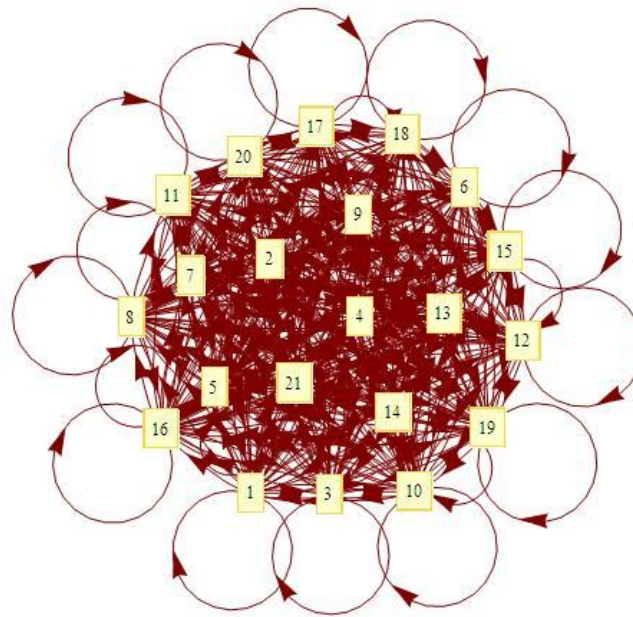


iniciales del HMM, en cambio los HMMs generados con programación evolutiva no requieren de conocimiento previo y proporcionan mejores resultados.

El siguiente experimento consistió en evolucionar un HMM con los datos de la secuencia de observación de tráfico de red de entrada y salida de la red. Los datos corresponden a 7 días de la semana con lectura del tráfico de red de cada hora, dando como resultado una secuencia de 168 datos en total.

El HMM multivariado (HCMMCM) evolucionado que se obtuvo con la observación multivariada de la dinámica del sistema, presenta 21 estados y se muestra en la Figura

Figura 10. HMMCM evolucionado con el ancho de banda de entrada y salida

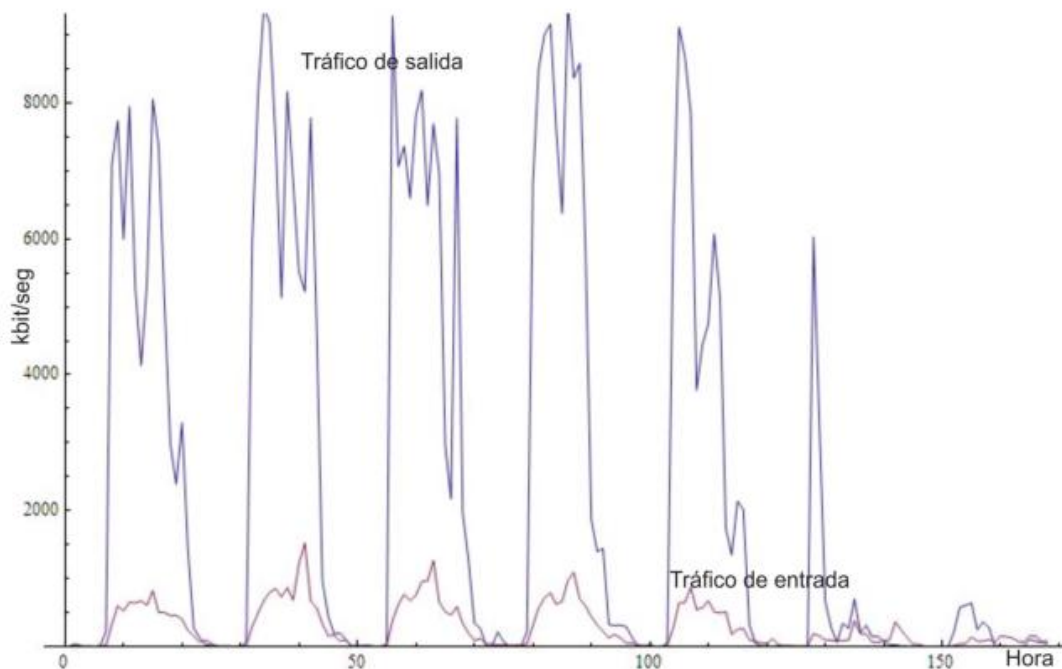


Fuente. Elaboración propia

El modelo HMMCM se utilizó para detectar las anomalías presentes en la secuencia de observación de los datos del consumo de ancho de banda real de la red. El comportamiento del consumo de ancho de banda de red con comportamiento anómalo se grafica en la figura 11; estos valores en los datos ocasionan un comportamiento distinto a la dinámica normal del sistema.

En el experimento hecho con el sistema detector de anomalías, basado en HMMCM, el umbral y la secuencia de observación anómala, detecto que cierto conjunto de valores de datos caían fuera del umbral o comportamiento normal de la dinámica del sistema, considerando a estos datos como probalisticamente anómalos. Es decir que sus valores de probabilidad presentaban una magnitud muy baja del comportamiento habitual del consumo de ancho de banda de la red.

Figura 11. Consumo real anómalo del ancho de banda de entrada y salida



Fuente. Elaboración Propia

En las pruebas hechas en el experimento se obtuvieron los siguientes resultados:

Con una ventana de 3 datos y umbral de 1.0×10^{-17} se detectaron 82 ventanas con datos anómalos de 166 ventanas en total. En la ventana de 4 datos y un umbral de 1.0×10^{-22} se detectaron 88 ventanas con anomalías de 165 ventanas en total. En las ventanas de 5 datos con un umbral de 1.0×10^{-33} se detectaron 100 ventanas que presentaban anomalías de un total de 163 ventanas. El resultado proporciona la conclusión de que entre más datos contenga la ventana, el valor de la probabilidad de la ventana se remarca, tanto como para caer dentro del umbral o fuera de este. En este experimento la ventana de 5 atos presento mejor rendimiento como consecuencia del tamaño de ventana y por la especificación del umbral.

El esquema de detección de anomalías fue capaz de detectar los datos anómalos sin dificultad gracias a la diferencia en el comportamiento de la dinámica del sistema modelado.

El esquema de detección de anomalías basado en HCMMCs, detecto que el tráfico de la red presentaba desviaciones considerables en su comportamiento normal.



CAPÍTULO V: PROPUESTA DE INVESTIGACION

En los últimos años se han propuestos varias arquitecturas de AIRSs, así como diferentes taxonomías de sistemas de respuesta automática frente a intrusiones, como por ejemplo las propuestas por **(Stakhanova, Basu, & Wong, 2007)** , **(SHAMELI-SENDI, EZZATI-JIVAN, JABBARIFAR, & DAGENAIS, 2012)** y **(Mateos Lanchas, 2013)**. De acuerdo a las taxonomías propuestas por estos tres trabajos de investigación, los AIRSs se pueden clasificar en función de las siguientes dimensiones:

- ✓ Tiempo de respuesta: estático o proactivo
- ✓ Según capacidad de adaptación: estático y adaptativo
- ✓ Según mecanismos de selección de la respuesta: estático, dinámico, sensible al coste de las respuestas.
- ✓ Según el modelo de evaluación del coste de la respuesta: modelo estático, modelo de evaluación estática y modelo de evaluación dinámica.
- ✓ Rapidez de reacción: lento y rápido.
- ✓ Capacidad de coherencia semántica.

Teniendo en cuenta las seis dimensiones en funciones de las cuales se puede clasificar un AIRS, lo ideal sería que este fuera: proactivo, adaptativo, sensible al coste de las respuestas y con un modelo de evaluación dinámica, rápido y semánticamente coherente.

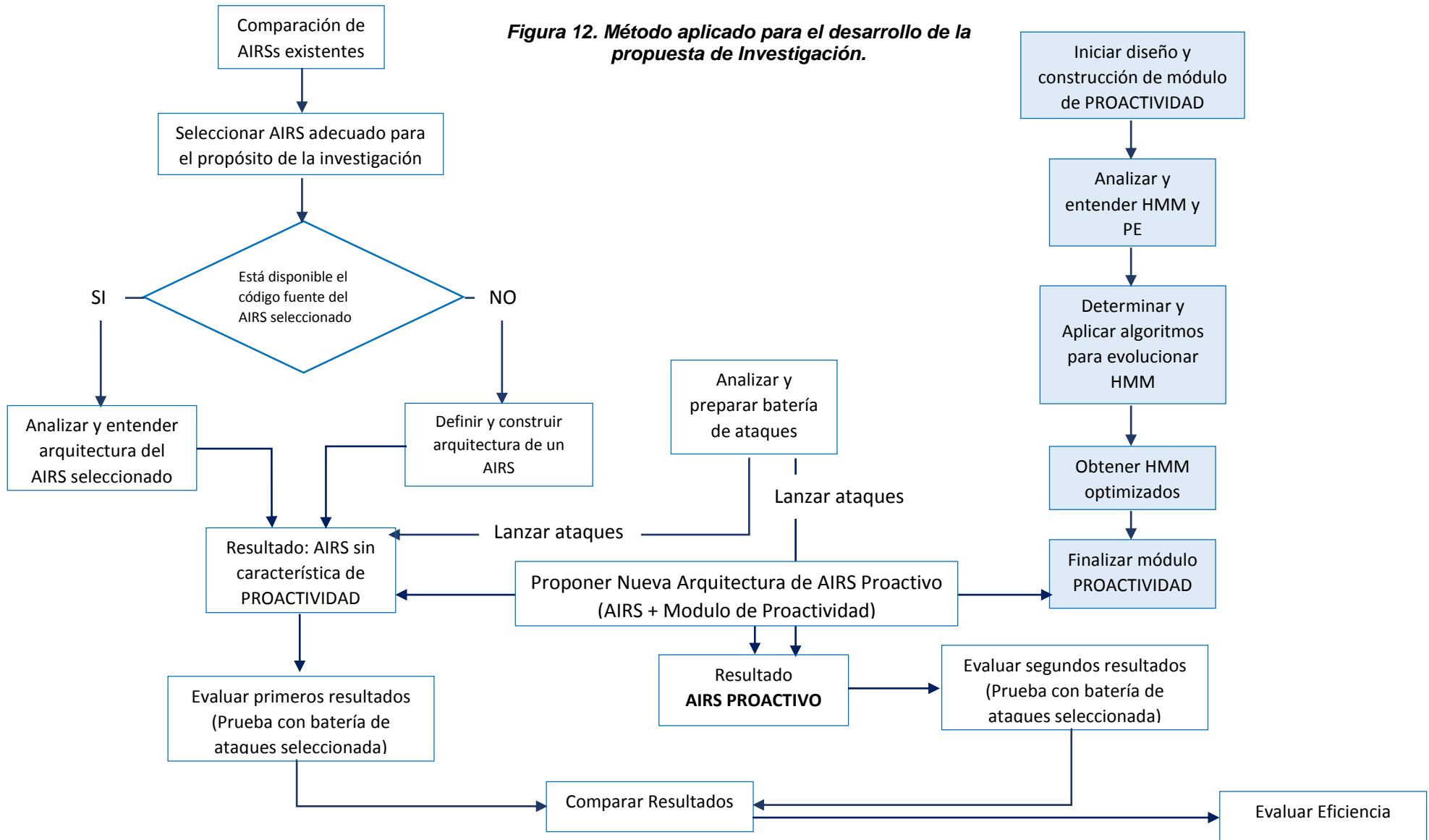
En base a las características mencionadas existen trabajos de investigación los cuales han tratado de llegar a cubrir las 6 características antes mencionadas, sin embargo ninguno ha llegado a incluirlas en su totalidad, para efectos de la presente investigación se tomara como base el trabajo realizado por **(Mateos Lanchas, 2013)**, denominado “Contribución a la Automatización de Sistemas de Respuesta Frente a Intrusiones mediante Ontologías”, ya que el AIRS propuesto tiene 5 de las características antes mencionadas, faltándole la característica de Proactividad, es aquí este trabajo de investigación buscara dotar de proactividad al AIRS propuesto por **(Mateos Lanchas, 2013)**.

Para llegar a objetivo buscado se plantea el método a seguir para el diseño e implementación de la propuesta de esta investigación. El método planteado se puede observar al detalle en la figura 13, en ella se detalla paso a paso actividades, técnicas y acciones secuenciales diseñadas y desarrolladas para conseguir el objetivo buscado; dotar de proactividad a un AIRS.

5.1 Método

La propuesta se desarrollara siguiendo el siguiente método:

Figura 12. Método aplicado para el desarrollo de la propuesta de Investigación.



Fuente. Elaboración Propia



A continuación se detallara paso a paso el método propuesto a la vez que se ira resaltando el cumplimiento de los objetivos específicos propuestos.

5.1.1 Comparación de AIRS existentes

El primer paso fue hacer un análisis comparativo de los AIRSs existentes, con el objetivo de plantear una serie de conceptos e ideas que se utilizaran a lo largo de este trabajo, resaltando las características que debería poseer de acuerdo a las descritas líneas arriba. Cada uno de los AIRSs estudiados posee una arquitectura, un mecanismo de inferencia de la respuesta óptima y unas métricas diferentes, aunque el objetivo a alcanzar sea un objetivo común; mitigar el ataque o reducir su impacto. En función de su arquitectura y principio de funcionamiento cumplen en mayor o menor medida las características deseables, pero ninguno aborda todas a la vez, como se observa en la siguiente tabla:

Tabla 10 Clasificación de los AIRSs existentes

	Rapidez de reacción	Adaptativo	Consideración del coste de las respuestas	Modelo de evaluación del coste	Coherencia semántica	Proactivo
CSM	-----	NO	NO	Estático	NO	NO
EMERALD	-----	NO	NO	Estático	NO	NO
AAIRS	-----	SI	NO	Estático	NO	NO
SARA	Lento	NO	-----	Estático	NO	NO
Network IRS	-----	NO	SI	Dinámico	NO	NO
Tanachaiwiwat's IRS	-----	NO	SI	Estático	NO	NO
ADEPTS	SI	SI	SI	Estático	NO	SI
MAIRF	-----	SI	NO	Estático	NO	SI
FAIR	-----	NO	SI	Estático	NO	NO
Stakhanova's IRS	-----	SI	SI	Estático	NO	SI
Jahnke	-----	NO	SI	Dinámico	NO	NO
IDAM & IRS	Lento	NO	SI	Estático	SI	NO
AIRS Basado en Ontologías	SI	SI	SI	Dinámico	SI	NO



Fuente (Mateos Lanchas, 2013), Actualizado por el Autor agregando AIRS Basado en Ontologías

En la tabla anterior se observa que ninguno de los AIRSs excepto ADEPTS y MAIRF aborda el problema de la proactividad en la respuesta a intrusiones.

ADEPTS, gracias a su árbol de caminos de intrusiones evita que los ataques evolucionen, por lo que se puede considerar que es un AIRS proactivo, de acuerdo a su flujo, el segundo paso en ADEPTS es calcular la probabilidad del siguiente nodo dentro del grafo de ataque, de esta forma el sistema se adelanta al ataque, pudiendo actuar en consecuencia. Respecto a MAIRF su funcionamiento es detectar el origen o fuente de la intrusión desde un host cualquiera del sistema una vez se ha producido el ataque de un activo de la red, con el fin de evitar que se propague la intrusión a otros puntos del sistema. Esta capacidad de prevención pone de manifiesto el comportamiento proactivo de MAIRF. Sin embargo aún no cumplen en su totalidad con las características deseables.

Debido a que ninguno de los AIRSs existentes satisface las seis características ideales de un AIRSs, y que ninguno de ellos aborda por completo la proactividad, se propone como trabajo de esta tesis una arquitectura de AIRS Proactivo, basado en Modelos Ocultos de Markov y Programación Evolutiva.

Además de satisfacer la propiedad de PROACTIVIDAD, la arquitectura propuesta trata de cumplir con el resto de requisitos.



5.1.2 Seleccionar el AIRS adecuado

Después de haber analizado y comparado los AIRSs existentes en el paso anterior, y de acuerdo a lo descrito en la Tabla 8, en función de su arquitectura y principio de funcionamiento el AIRS que cumplen en mayor medida las características deseables es el AIRS basado en Ontologías propuesto por (Mateos Lanchas, 2013). Ahora que ya se ha seleccionado el AIRS bajo el cual se trabajará y al cual se le agregará la característica de Proactividad, el siguiente paso era saber en qué medida se podía contar con la disponibilidad del código fuente del AIRS seleccionado a fin de modificarlo y agregarle la característica de Proactividad propuesta, así que lo siguiente fue tratar de establecer contacto con la autora del proyecto mencionado. Después de enviar diversos correos electrónicos a la autora de dicho proyecto no se obtuvo respuesta alguna, sin embargo al analizar al detalle el trabajo de investigación de la autora se pudo evidenciar que su trabajo estaba basado en un proyecto llamado RECLAMO (Virtual and Collaborative Honeynets based on Trust Management and Autonomous Systems applied to Intrusion Management), al igual que el trabajo de (Guaman Loachamin, 2013), ambos trabajos tienen su origen el proyecto RECLAMO, para beneplácito de esta investigación el código fuente de RECLAMO si está disponible a través de la siguiente web: <http://reclamo.inf.um.es/>, sin embargo surge una de las primeras limitaciones de esta investigación ya que si bien el código fuente está disponible, este se encuentra dividido en módulos los cuales no están integrados en una sola plataforma sino que trabajan de manera



independiente, por lo que ahora la dificultad estaba en cómo comunicar estos módulos para que puedan trabajar en conjunto. El paso a seguir fue establecer el contacto con los desarrolladores de RECLAMO a fin de poder obtener algún tipo de apoyo para el fin buscado, es así como se pudo establecer contacto con el PhD. Gregorio Martínez Perez y el Msc. Manuel Gil Perez dos de los desarrolladores del equipo del proyecto RECLAMO, quien en primera instancia respondieron al correo enviado, correos electrónicos que se adjuntan en el Anexo 3, y explico que el proyecto RECLAMO había llegado a su fin por falta de presupuesto económico por parte de las instituciones auspiciadoras y que no se había logrado desarrollar una plataforma que integrara todos los módulos por lo que se puso a disposición de la comunidad internacional para su uso y continuidad de desarrollo, tras explicar el trabajo que se pretendía desarrollar con la presente investigación, ambas personalidades ofrecieron su apoyo en el desarrollo e invitaron a formar parte del grupo que pretendía continuar con el desarrollo del proyecto RECLAMO.

5.1.3 Analizar y entender la arquitectura del AIRS seleccionado

Como se ha explicado en el apartado anterior no se pudo disponer del código fuente del AIRS basado en Ontologías pero se si pudo conseguir el código fuente del AIRs que dio origen al preseleccionado, me refiero al proyecto RECLAMO, el cual con ayuda del PhD. Gregorio Martínez Perez y el Msc. Manuel Gil Perez, se logró comprender su arquitectura y funcionamiento la cual se detalla a continuación.

RECLAMO (Virtual and Collaborative Honeynets based on Trust Management and Autonomous Systems applied to Intrusion Management)

El proyecto RECLAMO es un proyecto emprendido por la Universidad Politécnica de Madrid, propone una arquitectura de Sistema de Respuesta de Intrusiones Autónomo (AIRS). Este sistema infiere la respuesta más adecuada para un ataque determinado, teniendo en cuenta el tipo de ataque, información de contexto, y la confianza y reputación de los informes de intrusión de los IDS. RECLAMO propone un nuevo enfoque, desviar el ataque a una red trampa específica que es construido de forma dinámica en base a la información del ataque. Como se espera RECLAMO debe cumplir con ciertas características que definen a un IRS en general, como por ejemplo tener la capacidad de seleccionar una respuesta optima de entre un conjunto de respuestas recomendadas, en función del costo que implica ejecutarlo antes de que el objetivo se vea comprometido

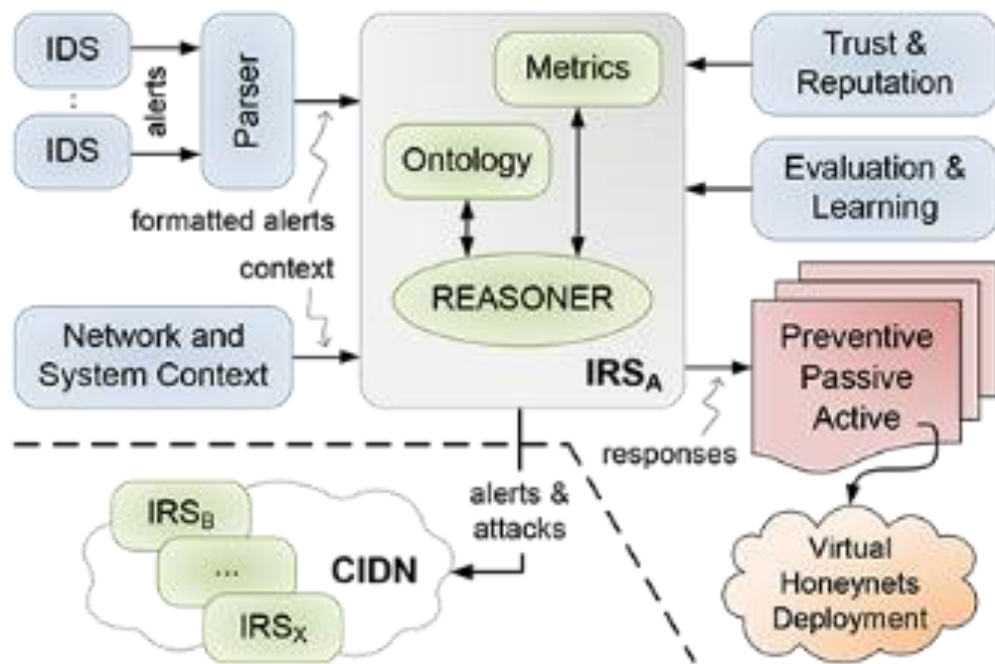


lo que de acuerdo a (Bishop, 2003) se conoce como Proactividad, la presente sección describe a RECLAMO desde un punto de vista práctico. Esto servirá de preámbulo para la sección donde se determinan las bases para la propuesta de la Arquitectura del AIRS Proactivo.

Arquitectura

La siguiente figura incluye una arquitectura prevista del Proyecto RECLAMO

Figura 13. Arquitectura de Proyecto RECLAMO



Fuente. (RECLAMO, 2013)



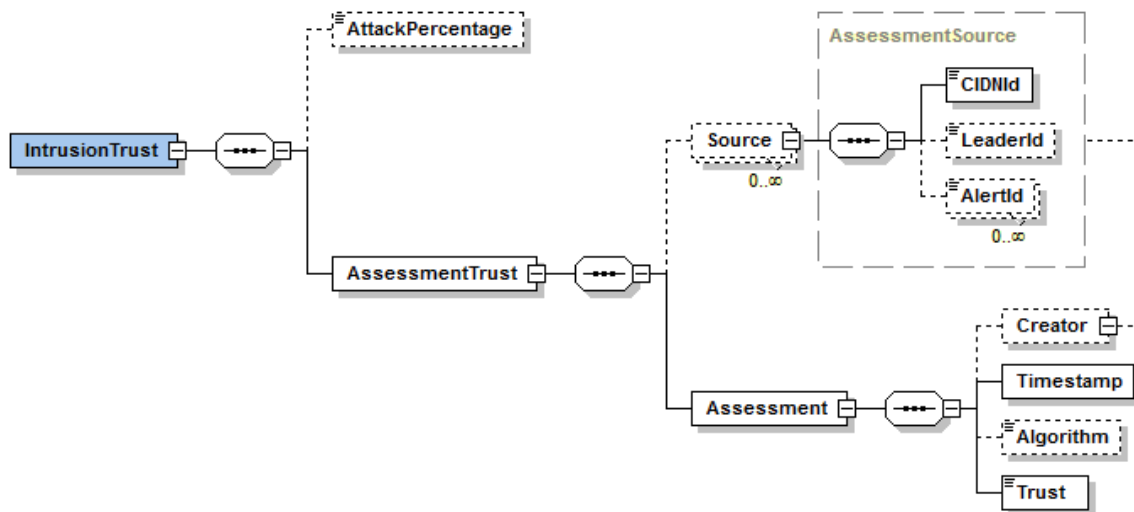
Formateador de Alertas

RECLAMO usa IDMEF (Formato de detección de intrusiones de mensajes de Exchange), definido por el IETF en el RFC 4765, como formato para la representación de eventos, ya que se adapta de forma correcta con los requisitos del sistema en cuanto a la información que deben ser compartidos entre los elementos de una red de colaboración de detección de intrusos. A pesar ello IDMEF no permite incluir información relacionada con la confianza del emisor de alertas, es decir si los IDS de donde provienen los reportes de alertas exponen un buen comportamiento al compartir alertas. Este hecho implica que IDMEF falla en determinar si las alertas vienen de los IDS tienen buena reputación, en este sentido se ha incluido una extensión XML al modelo de datos IDMEF, llamada IntrusionTrust, esta extensión se desarrolla como parte de la sección en el modelo de datos IDMEF con la finalidad de extenderla. La información incluida en IntrusionTrust incluye, además del valor de la confianza en sí mismo, la información sobre el algoritmo utilizado para calcular la puntuación de confianza de los emisores y la información acerca de la red de colaboración de detección de intrusos

La siguiente figura muestra la extensión IntrusionTrust, desde un punto de vista gráfico.



Figura 14. Representación grafica de la extensión "IntrusionTrust"



Fuente. (RECLAMO, 2013)

El primer elemento de alto nivel en esta extensión es AttackPercentage , que representa el porcentaje (entre 0 y 1) sobre el ataque en curso lleva a cabo hasta el momento. Es opcional, sólo se requiere este valor para una meta-alerta para indicar el porcentaje de un ataque de varios pasos. El segundo es el AssessmentTrust tipo complejo, que incluye el emisor (s) 'score confianza para una (meta) alerta calculada por el sistema de gestión de la confianza y la reputación propuesto en RECLAMO. Esta compuesto por:

AssessmentSource: información sobre la fuente (s) que emitió la evaluación, es decir, los nodos de computación las puntuaciones de confianza, sobre la IDS que generó la (meta) de alerta, a saber:

CIDNId: identificador CIDN a la que pertenecen los IDS.

LeaderId: identificador líder de la CIDN.



AlertId: identificadores de las alertas que fueron evaluados, especialmente útil para una alerta meta con posiblemente varias alertas individuales.

Assessment: información sobre la evaluación en sí, que está compuesto por tres subclases opcionales.

Creator: identificador de la entidad que finalmente creó la evaluación.

Time Stamp: fecha y hora en que se generó la evaluación.

Algorithm: algoritmo que fue utilizado por el sistema de la confianza y la gestión de la reputación para calcular la puntuación final de la confianza (meta) alerta, como RepCIDN , PeerTrust , Fire , PowerTrust o REGRET

Trust: Puntuación confianza final para el (meta) de alerta.

Contexto de Red

El propósito de este componente es generar una instantánea del tráfico de una red específica, y analizar los parámetros de tráfico, tales como fecha y hora actuales, tipo de protocolo, dirección IP de origen, puerto de origen, la dirección IP de destino, puerto de destino y el número de bytes recibidos y transmitidos. Entonces, este sistema calcula un parámetro llamado "Network Anomaly" mediante la comparación de la información instantánea anterior con un perfil de tráfico de red generado previamente. Este sistema puede ejecutar en dos modos diferentes: modo de aprendizaje para generar el perfil de las diferentes subredes y modo de detección de anomalías para calcular la anomalía de los indicadores de tráfico de la red en una subred específica en el tiempo de intrusión.

Contexto de Sistemas

El propósito de este componente es generar una instantánea de varios parámetros del sistema valores del componente atacado en el momento de la intrusión, como el número de procesos activos, uso de CPU, espacio libre en disco, la latencia, el número de usuarios conectados, el estado del sistema, el número de SSH falló inicios de sesión y número de procesos zombis. Este sistema puede ejecutar en dos modos diferentes: modo de aprendizaje para generar el perfil de todos los sistemas y modo de detección de anomalías para calcular la anomalía de los indicadores del contexto del sistema en un host específico en el tiempo de intrusión.

Sistema de Respuesta a Intrusiones Autónomo

RECLAMO ha propuesto y desarrollado un sistema de respuesta autónomo capaz de inferir la respuesta más apropiada para una intrusión dado, teniendo en cuenta no sólo la intrusión, sino también otros parámetros relacionados con él, como el contexto o la confianza y la reputación de el origen de la red. El sistema autónomo propuesto utiliza modelos de información, definidas formalmente con ontologías , para combinar los datos de intrusión, los parámetros de autoevaluación, la confianza y la reputación de los distintos elementos que intervienen, y la información de seguridad que viene de sistemas basados IDS-colaborativas en las mismas o diferentes dominios administrativos . Esta información de seguridad se evaluará con un conjunto de metricad de seguridad, representados con lenguajes de especificación comportamiento definidos formalmente, con el fin de razonar y deducir la



respuesta más adecuada, teniendo en cuenta todos los insumos y otros criterios especificados en las métricas de seguridad.

Métricas de Seguridad

En el proceso de respuesta a la intrusión, es necesario definir varias métricas que ofrecen un medio para medir y pesar hasta diferentes parámetros que son útiles para la selección de la respuesta, tales como la confianza IDS, el nivel de actividad de la red, la fiabilidad de los informes de intrusión, la importancia de los componentes de red, y la complejidad / gravedad / coste / eficacia de las respuestas. Estas métricas de seguridad especifican el comportamiento de la RECLAMO AIRs. RECLAMO propone el uso de ontologías y otras tecnologías de la Web Semántica, por lo que es necesario especificar las métricas de seguridad utilizando políticas idiomas o lenguajes de reglas, como SWRL. La especificación de la métrica de respuesta de una manera flexible y dinámica requiere el uso de un lenguaje específico capaz de expresar dichas métricas. RECLAMO hace uso de SWRL como un lenguaje formal para expresarlos. La principal ventaja de esta selección es la alineación con las tecnologías de la Web Semántica actuales que también están usando este lenguaje y el uso de herramientas y metodologías genéricas para el desarrollo del IRS.

Cuatro conjuntos de reglas se han definido en RECLAMO:

- ✓ Reglas para determinar el tipo de intrusiones basado en información de contexto.
- ✓ Reglas para obtener la fiabilidad alerta de intrusión.
- ✓ Reglas para inferir las respuestas recomendadas.
- ✓ Reglas para inferir la respuesta óptima.

Ontologías y lenguajes formales de definición del comportamiento

Coherencia semántica es la capacidad del sistema para entender la sintaxis y la semántica del informe de intrusiones con independencia de la fuente de la intrusión. El sistema de respuesta de intrusión entendería alertas o eventos de intrusión con diversas sintaxis de diferentes IDS, y sería capaz de determinar si dos alertas se refieren a los mismos o diferentes ataques. Por lo tanto, el sistema autónomo propuesto por RECLAMO clasifica coherencia semántica mediante el uso de ontologías como una tecnología de trabajo, como una de las principales ventajas en el uso de ontologías es la formalización de la semántica de la información. El uso de ontologías ayuda a apoyar la inclusión de diferentes y heterogéneos IDS, con diferentes formatos de intrusión y sintaxis, y diferentes fuentes de la red y el sistema de contexto.

RECLAMO ha adoptado OWL 2 (Lenguaje de Ontologías Web) como el lenguaje principal de ontología utilizado hoy en día en la Web Semántica para describir formalmente las definiciones de información. OWL 2 es un lenguaje de definición de conocimiento



que estructura la información en clases y propiedades (nominales o de relación entre objetos), con jerarquías y restricciones de rango y dominio.

Razonador

Este componente es uno de los principales componentes importantes de la arquitectura RECLAMO, es responsable de inferir la respuesta óptima para una intrusión dado. Teniendo en cuenta las políticas definidas previamente por el administrador del sistema, las instancias de la ontología que representan la información acerca de las respuestas, la información de contexto y alertas de intrusión, entre otros, el razonador ejecuta el proceso de inferencia para determinar la respuesta óptima a una intrusión específico, en un determinado lugar y tiempo. Hoy en día, hay varios razonadores semánticas que pueden ser utilizados como el núcleo de los AIRs. RECLAMO utiliza Pellet como razonador semántico.

Pero el componente Reasoner de la arquitectura RECLAMO no es sólo Pellet, es decir, además de Pellet, se ha desarrollado código de software adicional. Este componente es el responsable de la ejecución de una gran cantidad de funciones: comprobar la similitud o la continuación de una alerta de intrusión, invocan los módulos de contexto para el grado de anormalidad en un momento determinado, inferir las reacciones recomendadas y respuestas óptimas, obtener el nuevo conocimiento inferido, invocar el módulo Respuestas Ejecutor, invocar la evaluación y el aprendizaje del componente,

obtener información de la ontología usando Jena, actualizar la información inferida en la ontología RECAIRS y actualizar los nuevos resultados obtenidos en la respuesta histórica.

Confianza y Reputación

En RECLAMO, el concepto de la confianza y la reputación se aplica:

- ✓ IDS que se ejecuta en el mismo dominio de seguridad (modelo intra-dominio): dentro de una sola red de detección de intrusiones de Colaboración (CIDN).
- ✓ IDS que se ejecuta en diferentes áreas de dominio administrativos (modelo entre dominios): desde un punto de vista más global que saber lo que está sucediendo en toda la red.
- ✓ Un sistema de gestión de la reputación y la confianza se define en RECLAMO con el fin de mejorar la cobertura global de detección dejando caer las alarmas falsas o falsos positivos que surgen de nodos maliciosos o mal comportamiento. Este modelo permite a una CIDN detectar comportamientos maliciosos de acuerdo con la confiabilidad de los emisores de las alertas, calculados a partir de las interacciones previas con el sistema; aislar gradualmente tales IDS como su comportamiento empeora durante el tiempo.

El modelo de confianza y reputación propuesto en RECLAMO, llamado RepCIDN, ofrece una mayor precisión en la detección de ataques distribuidos, ya que será capaz de evaluar si cualquier IDS exhibe un comportamiento malicioso. Las falsas alarmas generadas por malintencionado o vulnerable IDS se caen directamente y no se publicarán en el resto de los componentes del sistema, disminuyendo a su vez el resultado de reputación asignado a los emisores (IDS) de tales falsas alertas.

Evaluación y Aprendizaje

Una vez que el sistema infiere un conjunto de respuestas óptimas, y esas respuestas se ejecutan, es necesario evaluar el resultado o el éxito de aquellos, con el fin de obtener retroalimentación para futuros procesos de inferencia. Esta evaluación también permite a los AIRSs aprender con el fin de ser más eficientes en las ejecuciones y las inferencias futuras. Este concepto de la evaluación y el aprendizaje está directamente relacionado con la capacidad de adaptación de los AIRS. La adaptabilidad es una característica de gran alcance que le permite al sistema ajustar dinámicamente la selección de la respuesta a los cambios del entorno durante el tiempo de ataque y modificar la respuesta elegida en función de otros factores externos, como el éxito y la tasa de fracaso de las respuestas previamente activadas.

RECLAMO propone el uso de anomalía en los indicadores de contexto para obtener la tasa de éxito de las reacciones inferidas, porque si una respuesta es eficaz o no, se puede reflejar en el sistema de anomalías y de red anomalía.

Dos enfoques diferentes han sido diseñados para gestionar el contexto de información anomalía para obtener la eficacia de la respuesta:

Enfoque basado en el uso del concepto de la Entropía Cruzada de Renyi y el enfoque basado en el uso de Redes Neuronales Artificiales (RNA), que forman parte de la máquina de aprendizaje.

Variación de la Entropía Cruzada de Renyi

Este enfoque se basa en el concepto bien conocido de entropía. La metodología que se incluye en RECLAMO aplica el concepto de variación de entropía Cruzada de Renyi para calcular la eficacia de la respuesta. La metodología de evaluación propuesta que el sistema sigue se divide en cuatro pasos:

- ✓ Calcular contexto varianza de entropía, mediante la comparación y el análisis de la red de anomalía y anomalías del sistema capturado antes y después de la ejecución de la respuesta. Iluminados por la definición de la entropía en teoría de la información,

se propone un concepto llamado "Contexto de variación de entropía", para modelar el contexto Parámetros cambios.

- ✓ Establecer umbrales: para obtener los umbrales, se requieren dos conjuntos de entrenamiento hechos de varias intrusiones para cada respuesta. Estos umbrales se establecen antes de ejecutar el AIRS por primera vez. En tiempo de ejecución, se utilizan para comparar en tiempo real el contexto de la varianza de entropía con el umbral correspondiente.
- ✓ Conseguir el nivel de éxito: en este paso el sistema utiliza una función que mapea el contexto de la varianza de la entropía después de la ejecución de una respuesta a nivel de éxito con el fin de limitar la incertidumbre en el proceso de evaluación.
- ✓ Conseguir tasa de éxito o eficacia de la respuesta global.

Redes Neuronales Artificiales

En este caso la red neuronal artificial incluida en RECLAMO implementa el algoritmo Back-Propagation porque es el método más utilizado de aprendizaje supervisado. Esta Red Neuronal Artificial estima la eficacia de la respuesta mediante el uso de una red neuronal pre entrenada. En este caso los parámetros de entrada son valores del grado de



anomalía del contexto de red. Como resultado se obtiene un valor de salida llamado SuccessLevel y a continuación se calcula la eficacia de la respuesta del AIRS.

Red de Colaboración de Detección de Intrusos (CIDN)

El diseño de una red de colaboración de detección de intrusiones (CIDN) que se ha sido definido en RECLAMO con la intención de construir y compartir un conocimiento colectivo acerca de las alarmas aisladas para detectar de manera eficiente y con precisión los ataques distribuidos. Está estructurado de forma modular para proporcionar escalabilidad, robustez y rendimiento en tiempo real por agrupar los sistemas más cercanos autónomos de detección de intrusiones (IDS) en grupos. La cuota de los IDS hacia adentro y combinar sus conocimientos colectivos (principalmente alertas) uno del otro dentro del mismo dominio administrativo, mientras que un pequeño grupo de ellos, siendo cada uno el líder de su dominio de seguridad, información de acciones de alto nivel entre los dominios administrativos involucrados en la proceso de detección de colaboración.

Todas las alertas generadas y compartidas por persona IDS, ya sea en un local o un contexto de alto nivel, son evaluados previamente por un Wise Committee (WC) antes de ser compartido con el resto de la IDS (conocimiento intra-

dominio) interna o con otra administrativa dominios (interdominio de conocimiento). Después de este procedimiento, tanto el cómo los dominios administrativos IDS y no pueden publicar o compartir sus alertas sin el consentimiento previo de su WC, evitando así la propagación de las alertas generadas por malintencionado o vulnerable IDS.

Despliegue de Respuestas

El sistema de respuesta automática a intrusiones y ataques propuestos en RECLAMO se basa en la inclusión de las denominadas respuestas engaño: la generación dinámica y el despliegue de las redes trampa en la que se desvían los ataques para confinar y analizarlos, y así obtener la mayor cantidad de información posible de cada tipo de ataque. Estas redes trampa serán optimizados y adaptados para cada ataque, que se basa en tecnologías de virtualización que actúa como un componente clave para la mitigación de ataques y reaccionar automáticamente a la detección de intrusos.

Una red trampa es un conjunto de honeypot sistemas (servidores, clientes, routers, switches, etc.) preparados para ser atacados al mismo tiempo que el seguimiento del ataque, e interconectado siguiendo una topología de red similar a la red de producción a proteger. Dada la complejidad de poner en marcha una red trampa, técnicas



de virtualización se convierten en una herramienta esencial que facilita en gran medida su implementación y la administración. Su capacidad para crear múltiples sistemas lógicos sobre una única máquina física puede reducir drásticamente el número de sistemas físicos necesarios para crear una red trampa, permitiendo al mismo tiempo aumentar la complejidad de las redes trampa creados. Además, la virtualización también se utiliza en RECLAMO para aumentar la disponibilidad de los servicios objeto de ataques, por lo que el usuario final no percibir ninguna diferencia en el proceso de prestación de servicios.

Despliegue de Trampas Virtuales

La generación red trampa dinámica se lleva a cabo mediante el uso de técnicas avanzadas de virtualización capaces de generar redes trampa heterogéneos gran escala. Las herramientas de apoyo tales técnicas deben facilitar la definición de las redes trampa (topología, direcciones, tipos de sistemas, etc.), despliegue, monitorización, etc., ocultando toda complejidad al sistema utilizando este tipo de plataformas de virtualización subyacentes. Entre las herramientas disponibles, RECLAMO hace uso de las redes virtuales sobre Linux (VNX), desarrollado por el grupo de investigación de la UPM, que recientemente fue mejorado para adaptarlo al contexto de la seguridad y la creación de

redes trampa altos de interacción. Esta herramienta incluye la capacidad de desplegar un escenario de red virtual a través de un grupo de servidores, la mejora de la escalabilidad de la solución y permitiendo la creación de redes trampa muy complejo, incluso a través de infraestructuras de racimo distribuidos.

Sistema Ejecutor de Respuestas

Además de las respuestas de engaño, el sistema de respuesta automática de intrusos desarrollado en RECLAMO es capaz de inferir y ejecutar otras medidas de respuesta, que se dividen en los siguientes grupos: respuestas reactivas, las respuestas de protección y las respuestas de recuperación. Algunas de las acciones de respuesta son la redirección del tráfico de red, bloquear una IP de origen, y añadir una nueva regla de firewall para denegar la conexión de una fuente IP específica.

La arquitectura de RECLAMO incluye un módulo responsable de la ejecución real de estas reacciones, la ResponsesExecutor sistema. Este sistema permite a los AIRSs ejecutar reacciones remotas y locales, tales como la modificación de una norma de firewall.

5.1.4 AIRS sin Proactividad

Hasta este punto ya tenemos un AIRS seleccionado y se ha comprendido al detalle su arquitectura y funcionamiento en el apartado anterior, sin embargo se ha podido evidenciar que aún no cuenta con la característica deseable de proactividad. El paso a seguir ahora es implementar y poner en marcha los módulos correspondientes al AIRS RECLAMO para hacer las primeras pruebas aun sin la característica de Proactividad, y evaluarla de acuerdo a los indicadores propuestos para luego ser comparados con los resultados del AIRS Proactivo y evaluar la eficiencia de la propuesta en este trabajo.

Para efectos de la presente investigación se tomara algunos de los módulos del AIRS RECLAMO, dicha selección se explica en el apartado 5.2 donde se propone una arquitectura modificada de acuerdo al objetivo buscado y detallando mínimamente el módulo de proactividad como principal aporte de este trabajo.

5.1.5 Analizar y Preparar Batería de Ataques

En la siguiente tabla se mostrara un resumen de los principales ataques, detallando su funcionamiento, forma de detección y la acción de respuesta que se puede tomar al respecto. Dicha información ha sido tomada de **(Mateos Lanchas, 2013)**, adaptada y actualizada para los propósitos de la presente investigación. Además con este paso estamos dando cumplimiento al **Objetivo Específico “A”**.



Tabla 11. Análisis de los tipos de Ataques

Tipo de Ataque	Nombre de Ataque	Funcionamiento	Detección	Acciones de Respuesta
Information gathering attack	Scanning	<p>Este ataque aprovecha las vulnerabilidades del protocolo TCP/IP, que utiliza puertos para envío y recepción de datos siguiendo el modelo cliente/servidor. Para establecer, mantener y terminar una conexión se utilizan bits de control (SYN, ACK, FIN) que pasan desapercibidos, vulnerabilidad que es aprovechada por los atacantes, ya que los filtros no pueden determinar con qué fin se lanzan los bits de control.</p> <p>Tipos:</p> <p>TCP SYN scanning: averigua puertos TCP abiertos. El atacante envía un paquete SYN y espera la respuesta. Al recibir un SYN/ACK envía inmediatamente un RST para terminar la conexión y registra ese puerto como abierto.</p> <p>TCP FIN scanning: averigua puertos TCP cerrados. El atacante envía un paquete de FIN y espera respuesta. Si recibe un RST el puerto está cerrado; si no recibe respuesta, el puerto está abierto.</p> <p>Por razones de implementación del protocolo en el sistema Windows es el único que responde siempre a todas las peticiones, estén los puertos abiertos o cerrado, por lo cual no da pistas sobre que puertos hay abiertos. Los sistemas vulnerables son UNIX LINUX NOVELL.</p> <p>Las principales ventajas de esta técnica de ataque es que no necesita privilegios especiales y posee una gran velocidad.</p> <p>Su principal desventaja es la facilidad de detección.</p>	<p>Este tipo de ataques se detecta por el gran número de conexiones y mensajes de error que se observan para los servicios a los que se ha conseguido conectar el atacante e inmediatamente se ha desconectado, ya que el ataque TCP scanning nunca abre una sesión TCP completa.</p>	<p>Monitorizar la red en busca de paquetes SYN a puertos restringidos (R38, Respuesta Pasiva y Respuesta Proactiva).</p> <ul style="list-style-type: none"> - Bloqueo de mensajes salientes con origen el puerto X (puerto sobre el que se han detectado las conexiones y mensajes de error) al puerto Y (el del atacante; el puerto origen de lospaquetes SYN). (R33, Respuesta Activa de Protección y Respuesta Proactiva). - Desactivar/Desinstalar servicios innecesarios. (R21, Respuesta Proactiva). - Cambiar el puerto a uno más alto, para evitar los escaneos (R32, Respuesta Activa de Protección y Respuesta Proactiva). - Rastrear la conexión del atacante para recopilar información (R7, Respuesta Activa de Reacción).
	Sniffing	<p>Ataque pasivo cuyo objetivo es interceptar tráfico de red de manera pasiva, sin modificación del mismo.</p> <p>Las herramientas utilizadas para llevar a cabo el ataque son los denominados Packet Sniffers, programas que monitorizan los paquetes que circulan por la red, que pueden instalarse en cualquier equipo de la red.</p> <p>El ataque consiste en colocar la tarjeta de red en modo "promiscuo" que desactiva el filtro de verificación de direcciones lo que provoca que todos los paquetes enviados a la red lleguen a esta placa (y pueden ser analizados por el atacante).</p> <p>Actualmente existen Sniffers que mediante el análisis del tráfico capturado pueden capturar todo tipo de información específica, como passwords de un recurso compartido o de acceso a una cuenta, números de tarjetas de crédito, direcciones de e-mails entrantes y salientes, relaciones entre individuos, etc.</p>	<p>Para su detección, es recomendable realizar las siguientes acciones:</p> <ul style="list-style-type: none"> - Revisar la lista de programas en ejecución para detectar alguna anomalía. - Mirar la lista de programas que se inician automáticamente al encender el PC o las tareas programadas para buscar evidencias. - Detectar tarjetas o adaptadores de red en modo promiscuo mediante el ipconfig. (el modo promiscuo es el usado por los atacantes para instalar los sniffers). 	<p>Como acciones de respuesta se pueden llevar a cabo las siguientes:</p> <ul style="list-style-type: none"> - Cerrar el puerto X de la máquina atacada al que el atacante puede conectarse para recuperar los datos que el sniffer ha capturado, es decir, interrumpir la conexión de red (R31, Respuesta Activa de Protección y Respuesta Proactiva). - Matar el proceso resultado de la ejecución del sniffer. (R22, Respuesta Activa de Protección). - Eliminar el fichero donde se están grabando los datos, imposibilitando así su acceso por parte del atacante (R11, Respuesta Activa de Restauración). <p>NOTA: estas tres respuestas se pueden llevar a cabo si el sniffer no se ha ocultado de forma adecuada y es detectado por los IDSs del sistema, caso en el que las herramientas de monitorización muestran el proceso que se está ejecutando, el fichero donde se están grabando los datos y la conexión de red, lo que permitirá ejecutar la respuesta adecuada.</p> <p>Un ataque muy similar a sniffing pero que además de interceptar el tráfico de red, el atacante se descarga la información capturada a su propia máquina para analizarla de forma más exhaustiva, es el conocido como Snooping. Las acciones de detección y respuesta ante este tipo de ataque son idénticas a las indicadas para sniffing.</p>



Network attacks	<p style="text-align: center;">Spoofing</p> <p>Para la realización de este ataque se utilizan técnicas de suplantación de identidad. Además, para llevarlo a cabo se necesita poseer gran conocimiento del protocolo en el que se basa el ataque. En función de la entidad objetivo de la suplantación y el protocolo, existen diferentes tipos:</p> <ul style="list-style-type: none"> - IP-Spoofing: suplantación de IP. Consiste en sustituir la dirección IP origen de un paquete TCP/IP por la IP que se quiere suplantar; las respuestas irán dirigidas a la IP falsificada. El atacante genera paquetes (TCP/IP) con una dirección IP de destino que no es la de la máquina víctima, sino una de una maquina determinada que sostiene una "relación de confianza" con el objetivo. -DNS-Spoofing: suplantación de identidad por nombre de dominio. Consiste en suplantar el DNS legítimo y falsificar la relación "nombre de dominio – IP", de tal forma que cualquier consulta de resolución de nombre se resuelva con una dirección IP falsa de un cierto dominio DNS o viceversa. Esto se consigue falseando las entradas de la relación Nombre de dominio/IP de un servidor DNS, mediante alguna vulnerabilidad del servidor en concreto o por su confianza hacia servidores poco fiables. De esta manera, el atacante proporciona a los servidores que realizan las consultas direcciones IP o nombres de dominio falsos con fines maliciosos, redirigiéndoles hacia páginas incorrectas. Las entradas falseadas de un servidor DNS son susceptibles de infectar (envenenar) el cache DNS de otro servidor diferente (DNS Poisoning). -Web- Spoofing: suplantación de una página web legítima. Consiste en redirigir la conexión de la máquina víctima hacia otras páginas Web que suplantan a la original, con el objetivo de obtener información acerca de la víctima. La página web falsa actúa a modo de proxy, solicitando la información requerida por la víctima a cada servidor original. El atacante puede modificar cualquier información desde y hacia cualquier servidor que la víctima visite. La forma en la que la víctima accede a la página web suplantada, suele ser mediante ataques de phishing. Este ataque se realiza mediante una implantación de código el cual nos robará la información, y es difícilmente detectable. -ARP- Spoofing: suplantación de identidad mediante falsificación de tablas ARP. El ataque consiste en construir tramas de solicitud y respuesta ARP modificadas con el fin de falsear la tabla ARP (relación IP-MAC) haciendo que la máquina víctima envíe los paquetes a un host atacante en lugar de a su destino elegido. El protocolo ARP trabaja a nivel de enlace de datos de la torre OSI, por lo que esta técnica sólo puede ser utilizada en redes LAN o en cualquier caso en la parte de la red que queda antes del primer router. - TCP-Spoofing: ataque spoofing más común, que consiste en falsificar una conexión TCP. Se conoce como adivinación del número de secuencia, y se basa en la idea de que si un atacante puede predecir el número inicial de secuencia de la conexión TCP generado por la máquina destino, el atacante puede adoptar la identidad de máquina de confianza y establecer una conexión con la máquina destino. <p>Los ataques de spoofing afectan a diario el funcionamiento de las redes TCP/IP. Estos son empleados como parte de un ataque o individualmente. La sencillez de su explotación ha masificado su uso. Todos los servicios TCP/IP sufren estas afectaciones.</p>	<p>Es un ataque difícil de detectar y de evitar. En el caso de Web spoofing, quizá la mejor medida sea instalar algún plugin en el navegador que muestre en todo momento la IP del servidor visitado: si la IP nunca cambia al visitar diferentes páginas WEB significará que probablemente estemos sufriendo este tipo de ataque.</p>	<p>Como acciones de respuesta ante este tipo de ataques se pueden llevar a cabo:</p> <ul style="list-style-type: none"> - Configurar el router para que no admita el envío de paquetes con IP origen no perteneciente a una de las redes que administra. Filtrado de paquetes por IP origen (R30, Respuesta Activa de Protección y Respuesta Proactiva). - Filtrar direcciones en el cortafuegos. (R26 y R28, Respuesta Activa de Protección y Respuesta Proactiva). - Realización de backup periódicos de la información sensible de los sistemas, que podrá ser utilizada tras la consecución de un ataque como respuesta de recuperación. (R9 y R13, Respuesta Activa de Recuperación y Respuesta Proactiva). <p>Una buena estrategia antispoofing, es aquella que propone considerar todos los riesgos que coexisten en la red TCP/IP para protegerla contra un ataque de spoofing, como respuesta de prevención, teniendo en cuenta que todos los servicios esenciales son vulnerables al spoofing y que la protección es similar para todos los casos.</p> <p>Entre las acciones de prevención, protección y recuperación que constituyen la estrategia se encuentran:</p> <ul style="list-style-type: none"> - Filtrado de paquetes en el router externo del cortafuegos. Es imposible eliminar todos los paquetes que tengas direcciones falsas (spoofing IP), pero sí es posible reducir el número si se filtran los paquetes y se restringe el flujo de entrada y salida de la red. - Configuración segura de los servicios DNS, SMTP, etc. - Protección de las conexiones y los datos, para mantener la confidencialidad, autenticidad e integridad de la información. Por ejemplo, cifrar las conexiones o utilizar la firma digital. - Utilizar barreras o filtros de red para registrar todas las conexiones a un servidor determinado. - Utilizar sistemas proxies para la navegación. Con esto se pretende detectar los posibles ataques de spoofing WWW. - Instalar programas sniffers que permiten detectar la ocurrencia de algunos ataques. - Explotar de forma adecuada los logs de los servicios y sistemas operativos. La configuración de la generación de logs debe estar encaminada a almacenar la mayor cantidad de información útil posible. Debe registrarse por ejemplo: dirección IP origen de las conexiones, el nombre del usuario, horario, duración de la conexión y tamaño e identificación de la transferencia efectuada. - Educar a los usuarios, que representa una forma de defensa muy importante contra el spoofing WWW. - Segmentación del tráfico, con el objetivo de separar todo el tráfico de administración de los sistemas del de los usuarios comunes y del de Internet. - Utilizar herramientas de detección de vulnerabilidades. - Actualizar los parches de los sistemas operativos. - Ofrecer sólo los servicios imprescindibles y de manera segura.
-----------------	--	--	--



	<p>Hijacking</p> <p>Ataque que consiste en ocupar una identidad obteniendo acceso no autorizado a un sistema, cuenta, etc., con el objetivo de poder falsificar la identidad del usuario atacado. Este tipo de ataque se produce cuando el atacante consigue interceptar una sesión ya establecida. Es un ataque de apropiación de identidad, que atenta contra la autenticación de los usuarios. El atacante roba una conexión después de que el usuario ha superado con éxito el proceso de identificación ante el sistema. El ordenador desde el que se lanza el ataque ha de estar en alguna de las dos redes extremo de la conexión, o en la ruta entre ambas.</p>	<p>Es un ataque muy difícil de detectar.</p>	<p>Como acciones de respuesta que pueda desplegar el AIRS de forma automática no hay ninguna efectiva. En cuanto a recomendaciones para evitar este tipo de ataques, se recomienda cifrar la información. Es el método más seguro para protegerse contra este tipo de ataques.</p>
	<p>Web applications</p> <p>Hay gran cantidad de ataques a aplicaciones web; a continuación se definen brevemente los más comunes: - Ejecución de código de forma remota: el atacante ejecuta código en el servidor vulnerable y obtiene información almacenada en él. Este ataque explota las vulnerabilidades del sistema debidas a errores de programación. Es un ataque simple pero eficaz. - Inyección de código SQL o SQL Inyention (ataques a bases de datos): ataque que afecta directamente a las bases de datos de una aplicación. Este ataque consiste en insertar o inyectar código SQL malicioso dentro de código SQL, para alterar el funcionamiento normal y hacer que se ejecute el código "invasor" dentro del sistema atacado. Esta técnica de ataque se usa para explotar sitios web que construyen sentencias SQL a partir de entradas facilitadas por el usuario. Es muy común entre los atacantes y dependiendo de las medidas de seguridad que tenga el sistema, el impacto del ataque puede variar desde revelación de información básica a ejecución de código remoto y compromiso total del sistema, mediante la alteración parcial o total de las bases de datos del sistema comprometido. - Cross Site Scripting (XSS): ataque que consiste en explotar las vulnerabilidades del sistema de validación de HTML incrustado, mediante la inyección en páginas web de código JavaScript o en otro lenguaje script similar con fines maliciosos. Es posible encontrar una vulnerabilidad XSS en aplicaciones que tenga entre sus funciones presentar la información en un navegador web u otro contenedor de páginas web. Sin embargo, no se limita a sitios web disponibles en Internet, ya que puede haber aplicaciones locales vulnerables a XSS, o incluso el navegador en sí. XSS es un vector de ataque que puede ser utilizado para robar información delicada, secuestrar sesiones de usuario, y comprometer el navegador, subyugando la integridad del sistema. El uso de AJAX para ataques de XSS no es tan conocido, pero sí peligroso. Se basa en usar cualquier tipo de vulnerabilidad de XSS para introducir un objeto XMLHttpRequest y usarlo para enviar contenido POST, GET, sin conocimiento del usuario. - Envenenamiento de cookies: ataque que consiste en la modificación del valor de las cookies antes de enviarlas de vuelta al servidor. A día de hoy, la mayoría de los sitios web, sólo almacenan un identificador de sesión (un número único generado aleatoriamente usado para identificar la sesión de usuario) en la propia cookie, mientras que el resto de la información es almacenada en el servidor, eliminando en gran medida el riesgo de sufrir este tipo de ataques. - Suplantación de contenido: técnica de ataque utilizada para engañar al usuario haciéndole creer que cierto contenido que aparece en un sitio web es legítimo, cuando en realidad no lo es. - Otros ataques definidos en otras partes del documento: ataques de fuerza bruta, desbordamiento de buffer, denegación de servicio, etc. Hay gran cantidad de ataques contra aplicaciones Web.</p>		<p>Entre las acciones de respuesta que pueden llevarse a cabo ante este tipo de ataque se encuentran:- Utilizar cortafuegos de aplicaciones web (WAF) ya que los cortafuegos tradicionales trabajan en la capa de red y de transporte, y al dejar el puerto 80 abierto no ofrecen ninguna clase de protección frente a los ataques contra aplicaciones web. Los WAF pueden ser a nivel de host y a nivel de red (R27 y R29, Respuesta Activa de Protección y Respuesta Proactiva). - Reconfigurar o restaurar el cortafuegos de aplicación (R35, Respuesta Activa de Protección y Respuesta Activa de Recuperación). - Evitar conexiones a bases de datos como un superusuario o como administrador de la base de datos (R14, Respuesta Activa de Protección y Respuesta Proactiva). - Cerrar el servicio o aplicación que proporciona el servidor afectado (R19, Respuesta Activa de Protección). - Reiniciar el servicio o aplicación que proporciona el servidor afectado (R20, Respuesta Activa de Protección y Respuesta Activa de Recuperación). - Restaurar el contenido de una página web en caso de que el servidor web sea el objetivo del ataque (R13, Respuesta Activa de Recuperación).</p>



<p>BackDoors</p>	<p>Es una técnica de ataque que consiste en la utilización de trozos de código en un programa que permite a quien las conoce saltarse los métodos usuales de autenticación para realizar ciertas tareas. Puede establecerse por dos tipos de usuarios: un programador que desarrolla una aplicación y establece backdoors para acceder al programa más rápidamente en la fase de pruebas o por error; un atacante que establece una backdoor mediante otras técnicas de ataque como exploits o troyanos para tener acceso a la máquina víctima con fines maliciosos. Una puerta trasera no es peligrosa si es utilizada por usuarios no malintencionados, pero es una entrada al sistema no controlada, lo que la convierte en peligrosa para la seguridad e integridad del sistema.</p> <p>La mayoría de las puertas traseras se introducen en nuestro sistema en forma de troyanos cuya finalidad es siempre malévol. Por otra parte, para poder utilizar las puertas traseras debe existir un cliente y un servidor, pero los troyanos tienen la capacidad de instalar en el sistema un servidor para posteriormente acceder a él como cliente y tener control remoto.</p> <p>El hecho de tener una entrada no controlada al sistema, provoca un fallo de seguridad si se utiliza de forma malintencionada, ya que cualquiera que conozca el agujero o lo encuentre en su código podrá saltarse los mecanismos de control y autenticación habituales.</p> <p>Para abrir una puerta trasera en un sistema Unix a veces basta añadir dos líneas de texto.</p>	<p>Para detectar una backDoor hay que tener en cuenta los efectos que produce la instalación de ésta en el sistema. Por ejemplo, se puede saber que hay instalada una backdoor si se detecta la siguiente actividad no autorizada por el usuario o administrador del sistema:</p> <ul style="list-style-type: none"> - Monitorización del sistema completo - Descarga de virus. - Desinstalación e instalación de programas. - Borrado de información de usuario. - Manipulación del registro del sistema. 	<p>Entre las acciones de respuesta que pueden llevarse a cabo ante este tipo de ataque se encuentran:</p> <ul style="list-style-type: none"> - No aceptar peticiones ni llamadas que provengan de usuarios desconocidos o sospechosos (R24 y R25, Respuesta Activa de Protección y Respuesta Proactiva). - En el caso de borrado de información de usuario, restaurar el activo afectado a su versión anterior previa al ataque. (R13, Respuesta Activa de Recuperación). - Aumentar la sensibilidad del IDS para la detección de troyanos (R6, Respuesta Activa de Recuperación, Respuesta Pasiva y Respuesta Proactiva). - Restaurar el sistema a una versión correcta previa a la intrusión (R18, Respuesta Activa de Recuperación). - Identificar la backdoor y bloquearla, cerrarla (R31 y R33, Respuesta Activa de Protección y Respuesta Proactiva).
<p>Password Attacks</p>	<p>Guessing. Brute Force</p> <p>Ataque que consiste en generar e ir probando con todas las combinaciones de caracteres (letras, números, etc.) posibles hasta adivinar la contraseña (password). Es un proceso automatizado de prueba y error.</p> <p>Este tipo de técnica consiste en la obtención por "fuerza bruta" de aquellas claves que permiten acceder a los sistemas, aplicaciones, cuentas, etc. objetivos del ataque. Esta técnica es altamente eficaz ya que la mayoría de las contraseñas de acceso se pueden obtener fácilmente debido a la mala concienciación de los usuarios, que suelen elegir contraseñas fáciles, cortas y no las cambian con regularidad.</p> <p>Un ataque de fuerza bruta "pura" teóricamente no puede ser resistido por ningún sistema, siempre y cuando se disponga del tiempo suficiente y recursos suficientes. Será más complicado adivinar claves largas, pero si no existe limitación de tiempo, el atacante conseguirá su objetivo. Estas limitaciones físicas son dinámicas y van disminuyendo a medida que los ordenadores van alcanzando más capacidad de procesamiento.</p> <p>Como medida de protección, muchos sistemas niegan el acceso después de un determinado número de fallos de autenticación, lo que dificulta la labor de los atacantes que usan esta técnica.</p> <p>Por otra parte, esta técnica tiene la ventaja de que la mayoría de los ataques de fuerza bruta se hacen offline, es decir, se obtiene el archivo que contiene la lista cifrada de passwords y se trabaja desde la propia máquina del atacante, sin necesidad de mantener conexión con la máquina víctima.</p>	<p>Como técnicas de detección de ataques de adivinación por fuera bruta se tienen:</p> <ul style="list-style-type: none"> - Instalar en los sistemas de la organización alguna herramienta de detección de ataques de fuerza bruta, como por ejemplo Denyhosts, blocksshd, fail2ban, Brute Force Detection, etc., que informan en cada momento de los intentos no autorizados y todo tipo de actividad de autenticación sospechosa que se ejecute contra los servidores, en especial contra el servidor SSH. - Mirar los logs de autenticación del sistema. - Ataque fácil de detectar por la mayoría de los cortafuegos (un gran número de intentos de conexión desde una sola dirección IP). No obstante, desde la utilización de redes botnets para la masificación de ataques, es más difícil su detección. 	<p>Entre las acciones de respuesta que pueden llevarse a cabo ante este tipo de ataque se encuentran:</p> <ul style="list-style-type: none"> - Añadir una entrada que deniegue el acceso al usuario atacante detectado, (R14, Respuesta Activa de Protección y Respuesta Proactiva). - Añadir reglas a la configuración de las tablas IP de la máquina atacada que impidan cualquier tipo de tráfico entre el atacante y la máquina atacada. Bloquear la IP o servicio implicado. (R31, Respuesta Activa de Protección y Respuesta Proactiva). - Hacer un whois de la IP y rastrear la conexión del atacante para recopilar información (R7, Respuesta Activa de Reacción). - No permitir acceso a root en sistemas Linux (R14, Respuesta Activa de Protección y Respuesta Proactiva). - Restaurar los activos afectados (modificados, eliminados, etc.) con motivo de la intrusión (R13 y R11, Respuesta Activa de Recuperación).



	<p>Guessing. Dictionary attack</p>	<p>Ataque similar al anterior, que consiste en probar con todas (o la mayoría) de las palabras conocidas en un idioma dado: un buen diccionario tiene entre 100 mil y 200 mil palabras y es un excelente punto de partida. No es lo mismo que un ataque de pura fuerza bruta, que prueba todas las ombinaciones posibles.</p> <p>Como se indica en, "Los diccionarios son archivos con millones de palabras, las cuales pueden ser passwords utilizadas por los usuarios". El atacante utiliza un programa que se encarga de probar cada una de las palabras cifrando cada una de ellas (mediante el algoritmo utilizado por el sistema atacado) y comparando la palabra cifrada contra el archivo de passwords del sistema atacado (previamente obtenido). Si coinciden se ha encontrado la clave de acceso al sistema mediante el usuario correspondiente a la clave hallada. Actualmente es posible encontrar diccionarios de gran tamaño orientados, incluso, a un área específica de acuerdo al tipo de organización que se esté atacando.</p> <p>Los ataques de diccionario tienen pocas probabilidades de éxito con sistemas que utilizan contraseñas fuertes con letras en mayúsculas y minúsculas mezcladas con números y con cualquier otro tipo de símbolos. Sin embargo para la mayoría de usuarios recordar contraseñas tan complejas resulta complicado, por lo que un ataque de diccionario suele ser más eficaz que uno de fuerza bruta.</p>	<p>Los mecanismos de detección de esta técnica de ataque de adivinación son similares a los especificados para Guessin - Brute Force.</p>	<p>Las acciones de respuesta de esta técnica de ataque de adivinación son similares a los especificados para Guessin - Brute Force</p>
	<p>Exploits</p>	<p>Ataque que consiste en utilizar un trozo de código, fragmento de datos o secuencia de comandos y/o acciones llamado exploit, para explotar los agujeros o vulnerabilidades de seguridad existentes en un sistema de información, con el objetivo de conseguir un acceso no autorizado a la máquina objetivo o provocar un comportamiento no deseado de dicho sistema o de alguno de los servicios que éste presta. Los objetivos perseguidos por los atacantes mediante la explotación de exploits son muchos y variados, como por ejemplo, conseguir acceso al sistema de forma no autorizada, obtener privilegios no concedidos de forma lícita, realizar un ataque de denegación de servicio, etc. Cabe aclarar que el término exploit hace referencia al trozo de código que el atacante ejecuta para explotar la vulnerabilidad encontrada y conseguir su objetivo. Para llevar a cabo esta técnica de ataque, los atacantes ejecutan programas que permiten explotar estos "agujeros" de seguridad que reciben el nombre de Exploits. El funcionamiento es el siguiente: el programa aprovecha la debilidad, fallo o error hallado en el sistema (hardware o software) para conseguir acceso al mismo. Nuevos exploits (que explotan nuevas vulnerabilidades en los sistemas) se publican cada día por lo que mantenerse informado de los mismos y de las herramientas para combatirlos es esencial.</p>	<p>En máquinas Windows hay ciertos indicios que permiten detectar que un atacante ha explotado una vulnerabilidad mediante un exploit: - El tráfico de red de salida es sospechosamente alto, sobre todo cuando el ordenador está inactivo o no necesariamente cargando datos. - Reducción drástica del espacio libre en disco, o existencia de archivos de aspecto sospechoso en los directorios raíces de cualquiera de los discos. Después de penetrar en el sistema, muchos hackers realizan un escaneo masivo para encontrar documentos interesantes o archivos que contengan contraseñas. - Se produce un gran aumento en el número de paquetes que son bloqueados por un cortafuegos personal, procedentes de una dirección simple. De la misma forma, posibles indicios de exploits en máquinas Unix son:- Existencia de archivos con nombres sospechosos en el archivo /tmp folder.- Instalación de backdoors o modificación de utilidades estándar del sistema que se usan para conectarse con otros sistemas.- Alteración de los ficheros /etc/passwd, /etc/shadow, u otros archivos de sistemas en el directorio /etc, por un usuario sin privilegios. A veces los atacantes pueden añadir un nuevo usuario en el fichero /etc/passwd, para que este tenga acceso remoto al sistema de forma lícita en una fecha posterior. - Modificación del fichero /etc/services así como de /etc/inet.conf, para establecer una backdoor en puerto sospechoso o no usado.</p>	<p>Entre las acciones de respuesta que puede llevar a cabo el AIRS ante este tipo de ataque se encuentran: - Agregar o borrar reglas de filtrado en los cortafuegos, (R26, Respuesta Activa de Protección y Respuesta Proactiva). - Bloquear la IP del atacante temporalmente hasta que cesen los intentos de conexión (R31, Respuesta Activa de Protección y Respuesta Proactiva). - Terminar conexión TCP establecida por el atacante (R34, Respuesta Activa de Protección y Respuesta Proactiva). - Bloquear conexiones de red entrantes y salientes sospechosas (R33, Respuesta Activa de Protección y Respuesta Proactiva). - Eliminar cualquier nombre de usuario sospechoso en el archivo de contraseñas (R15, Respuesta Activa de Protección y Respuesta Proactiva).</p>



<p>Virus</p>	<p>Un virus es un programa que se replica a sí mismo y que se propaga a través de archivos infectados. Una vez cargado en memoria cumple las instrucciones programadas por su creador, que pueden ser: destruir ficheros, modificar ficheros, sobrecargar recursos, entre otros. Un virus es un trozo de código incrustado en un programa que se replica a sí mismo y se propaga a través de archivos infectados, cuyo objetivo es alterar el funcionamiento normal de la máquina atacada, creando o modificando ficheros, sobrecargando los recursos del sistema, etc. Un virus tiene las siguientes características: - Se replican (propagan) y ejecutan por sí mismos. - Reemplazan archivos ejecutables por otros infectados con el código de éste. - Contienen una carga dañina (payload) con distintos objetivos: destruir datos dañando el sistema; bloquear las redes informáticas generando tráfico inútil; ser "algo molestos". Su mecanismo de funcionamiento es el siguiente: - Se ejecuta un programa infectado (por desconocimiento del usuario). - El código del virus queda residente en RAM, aun cuando el programa que lo contenía haya terminado. - El virus toma el control de los servicios básicos del SO, infectando archivos ejecutables que serán llamados posteriormente. - Finalmente, se añade el código del virus al del programa infectado y se graba en disco, con lo cual el proceso de replicación se completa. Los efectos de un virus varían desde la simple visualización de una pelota de ping pong rebotando por toda la pantalla de escritorio de la máquina víctima, hasta la eliminación de ficheros o desinstalación de programas.</p>	<p>Como técnicas de detección de este tipo de ataques se distinguen: - Uso de programas antivirus que analizan la firma del virus. - Cambios producidos en la organización del sistema de ficheros o en la información del sistema. - Ejecución del método heurístico, que implica el análisis del comportamiento de las aplicaciones para detectar una actividad similar a la de un virus conocido. - Aumento del consumo de recursos del sistema que implican una pérdida de productividad, cortes en los sistemas de información o daños a nivel de datos.</p>	<p>Entre las acciones de respuesta que puede llevar a cabo el AIRS ante este tipo de ataque se encuentran: - Analizar las trazas que ha dejado un virus para eliminarlo una vez detectado (R7, Respuesta Activa de Reacción). - Generar filtros de ficheros dañinos, por ejemplo en el sistema de correo. - Realizar copias de seguridad de todos los activos del sistema (R9, Respuesta Proactiva) con el fin de futuras restauraciones (R13, Respuesta Activa de Recuperación). - Restauración completa de la máquina o sistema comprometido (formateo) (R18, Respuesta Activa de Recuperación). - Denegar el acceso de forma selectiva o completa a un archivo (R10, Respuesta Activa de Protección y Respuesta Proactiva).</p>
<p>Worms (Gusanos)</p>	<p>Es un programa que se reproduce por sí mismo, que puede viajar a través de redes utilizando los mecanismos de éstas y que no requiere respaldo de software o hardware (como un disco duro, un programa host, un archivo, etc.) para difundirse. Un gusano puede considerarse una técnica de ataque similar a los virus pero que no dependen de archivos portadores para poder llegar e infectar otros sistemas. Las principales características de los gusanos son: - No altera los archivos del sistema sino que residen en la memoria y se duplica a sí mismo. - Pueden modificar el SO con el fin de autoejecutarse como parte del proceso de inicialización del sistema. - Utilizan las partes automáticas de un SO que generalmente son invisibles al usuario. Principalmente su mecanismo de funcionamiento es el siguiente: explotar vulnerabilidades de la máquina objetivo o utilizar algún tipo de ingeniería social para engañar a los usuarios y poderse ejecutar. Los gusanos actuales se propagan principalmente con clientes de correo electrónico (en especial de Outlook) mediante el uso de adjuntos que contienen instrucciones para recolectar todas las direcciones de correo electrónico de la libreta de direcciones y enviar copias de ellos mismos a todos los destinatarios. Generalmente, estos gusanos son scripts (típicamente en VBScript) o archivos ejecutables enviados como un adjunto, que se activan cuando el destinatario hace clic en el adjunto.</p>	<p>Como técnicas de detección de este tipo de ataques se distinguen: - Aumento drástico del consumo de recursos de la máquina víctima. Tienen una incontrolada replicación, por lo que consumen muchos recursos del sistema, hasta el punto de que los procesos del sistema se ejecutan de forma excesivamente lenta o no pueden ejecutarse. - Modificación de información del sistema de forma no autorizada. - Ver Virus. Detección.</p>	<p>Entre las acciones de respuesta que puede llevar a cabo el AIRS ante este tipo de ataque se encuentran: - Similar a los virus.</p>



<p>Troyanos</p>	<p>Este ataque se basa en la utilización de un código malicioso que se encuentra en un programa sano (por ejemplo, un comando falso para crear una lista de archivos que los destruye en lugar de mostrar la lista). El atacante introduce el trozo de código malicioso dentro de un programa útil del sistema. La ejecución del programa origina la ejecución de la rutina, instalando así el troyano en el sistema al ejecutar dicho archivo. El objetivo de este ataque no es reproducirse para infectar a otras máquinas, sino abrir un puerto en la máquina víctima para que un atacante pueda establecer conexión con ella a través del puerto o backdoor establecida, y tener control sobre la máquina comprometida. Para ello, debe infectar la máquina obligando a abrir un archivo infectado que contiene el Troyano, y luego, acceder a la máquina a través del puerto abierto. Hay una serie de puertos fijos utilizados generalmente por los troyanos, como por ejemplo, el puerto 28678 utilizado por Exploiter, el puerto 29104 utilizado por NetTrojan, puerto 29369 utilizado por ovasOn, etc.</p> <p>Una vez instalado, el troyano puede realizar las siguientes tareas:</p> <ul style="list-style-type: none"> - Espiar al usuario: obtención de contraseñas mediante captura de las pulsaciones del teclado. - Realizar operaciones maliciosas dentro de la máquina como la instalación de programas. - Crear de un agujero de seguridad para que los usuarios externos accedan a áreas protegidas. - Establecimiento de BackDoors, el atacante puede acceder a la máquina comprometida. 	<p>La infección es evidente por los siguientes síntomas:</p> <ul style="list-style-type: none"> - Actividad anormal de dispositivos de red o del propio sistema, como por ejemplo: § Reacciones extrañas de los dispositivos periféricos como el ratón o el teclado. § Programas que se abren en forma inesperada. § Bloqueos repetidos. 	<p>Entre las acciones de respuesta :</p> <ul style="list-style-type: none"> - Agregar o borrar reglas de filtrado sobre el cortafuegos (R26 y R28, Respuesta Activa de Protección y Respuesta Proactiva), que bloqueen el establecimiento de conexiones salientes por parte de programas cuyo origen se desconoce (R33, Respuesta Activa de Protección y Respuesta Proactiva). - Eliminar el troyano, mediante herramientas como The Cleaner (R11, Respuesta Activa de Recuperación) - Formatear la máquina afectada (R18, Respuesta Activa de Protección y Respuesta Proactiva). - Cerrar el puerto utilizado por el troyano para abrir una backdoor (R31 y R34, Respuesta Activa de Protección y Respuesta Proactiva). - Realizar copias de seguridad de todos los ficheros del sistema (R9, Respuesta Proactiva). - Rastrear la conexión del atacante para recopilar información (R7, Respuesta Activa de Reacción). - Denegar el acceso de forma selectiva o completa a un archivo (R10, Respuesta Activa de Protección y Respuesta Proactiva).
<p>Buffer Overflow</p>	<p>Ataque que se basa en la ejecución de un código arbitrario en un programa al enviar un caudal de datos mayor que el que puede recibir. Un programa que admite datos de entrada con parámetros, los almacena temporalmente en una zona de la memoria denominada búfer. El problema es que algunas funciones no admiten este tipo de desbordamiento y provocan el bloqueo de la aplicación, lo que permite la ejecución del código arbitrario del atacante y permite el acceso al sistema. Los pasos que rigen el comportamiento habitual de este ataque son los siguientes: El servidor reserva unas posiciones de memoria para almacenar los datos que le llegan por la red. Para estimar el tamaño de la memoria a reservar, el programador comprueba cuál es el tamaño máximo de los datos que le pueden llegar según el estándar del protocolo implementado. En principio, se supone que no deberían llegar datos de tamaño superior al máximo permitido por el estándar. - Un atacante genera un paquete de datos malicioso con un tamaño superior al máximo permitido. - El servidor recibirá esos datos y los irá almacenando en la memoria reservada. Cuando el servidor llena el espacio reservado se pueden dar dos situaciones: El servidor comprueba la situación y advierte que el tamaño de los datos enviados supera el máximo permitido, por lo que rechaza la petición por no ser conforme al estándar. En este caso, no se da opción a la realización de un ataque. El servidor no comprueba la situación y sigue transfiriendo los datos a memoria, sin darse cuenta de que los está enviando a posiciones de memoria que no han sido reservadas y que, por lo tanto, está sobrescribiendo posiciones de memoria. Es en este caso, cuando se puede intentar realizar el ataque. - Si el servidor no comprueba el desbordamiento de buffer, el comportamiento más normal es intentar sobrescribir posiciones de memoria protegidas por el sistema operativo, caso en que el propio sistema operativo aborta la ejecución del servidor con un programa de error. No obstante, existe un caso en el que se sobrescriben posiciones de memoria especiales, las correspondientes a la pila del sistema operativo, que contienen las direcciones de las instrucciones que el sistema operativo va a ejecutar. - Si el atacante consigue que uno de los datos que se desborda del buffer sobrescriba la pila del sistema operativo, puede forzar la escritura de la dirección del buffer de memoria donde el servidor ha ido almacenando los datos que le ha enviado previamente el atacante. - Por último, el sistema operativo intentará ejecutar el código apuntado desde la pila, es decir, intentará ejecutar el contenido de los datos que el atacante ha enviado previamente. Si estos datos contienen instrucciones que permiten al atacante hacerse con el control del sistema, el sistema operativo lo ejecutará y el atacante habrá realizado con éxito un acceso remoto no autorizado al sistema.</p>	<p>Técnica de Detección</p> <p>Como técnicas de detección de este tipo de ataques se distinguen:- El sistema ejecuta procesos o acciones no ordenadas por el administrador o el usuario del sistema.- Modificación de información del sistema.- Aumento del consumo de disco en momentos no esperados.</p> <p>Acciones de Respuesta</p> <p>Entre las acciones de respuesta que puede llevar a cabo el AIRS ante este tipo de ataque se encuentran: - Eliminar el proceso que se está ejecutando con el trozo de código malicioso (R22 y R23, Respuesta Activa de Protección y Respuesta Activa de Recuperación. - Devolver los activos afectados a su estado previo al ataque (R13 y R18 Respuesta Activa de Recuperación).</p>	



<p>Denial of Service</p>	<p>Ataque cuyo objetivo es saturar los recursos de la máquina víctima de tal forma que se inhabilitan los servicios prestados por la misma durante un periodo de tiempo indefinido. Hace inaccesible un servicio o recurso a los usuarios legítimos, provocando normalmente, la pérdida de la conectividad de la red por el consumo del ancho de banda de la red del sistema atacado o sobrecarga de los recursos computacionales del mismo.</p> <p>El ataque se puede dar de muchas formas, pero todas utilizan el protocolo TCP/IP para llevarse a cabo. Se genera mediante la saturación de los puertos con flujo de información, haciendo que el servidor se sobrecargue y no pueda seguir prestando servicios a los usuarios habituales, o el sistema se vuelva inestable.</p> <p>Como evolución del DoS, ha surgido el ataque DDoS (Distributed Denial of Service), una ampliación del ataque DoS, que se lleva a cabo instalando varios agentes remotos en muchos ordenadores localizados en diferentes puntos (o en el mismo). El invasor consigue coordinar los agentes para incrementar de forma masiva la saturación de información, hasta el punto de conseguir lanzar un ataque de cientos de máquinas dirigido a una máquina o sistema objetivo. Esta técnica se ha revelado como una de las más eficaces y sencillas a la hora de colapsar servidores.</p> <p>El objetivo de ambos ataques no reside en recuperar, modificar o destruir datos, sino que se trata de un ataque a la imagen y reputación de las compañías afectadas. Puede ser que los atacantes necesiten que un sistema caiga para que un administrador lo reinicie. Es muy fácil vulnerar un sistema justo durante el reinicio, antes de que todos los servicios estén totalmente operativos. No se trata de ataques muy complicados pero son ataques eficaces contra cualquier tipo de equipo o sistema operativo.</p>	<p>Como indicios para detectar este tipo de ataques se distinguen:</p> <ul style="list-style-type: none"> - Aumento del número de conexiones fallidas o semiabiertas. - Consumo de disco no correspondiente a la actividad de un usuario legítimo. - Caída inexplicable de un sistema conectado a la red. - Fallo general del sistema o cuelgue de procesos por falta de CPU que está siendo utilizada por el atacante. - Redirección del tráfico de la máquina víctima hacia un agujero negro. (Ataques a los DNS y de enrutamiento). 	<p>Entre las acciones de respuesta que puede llevar a cabo el AIRS ante este tipo de ataque se encuentran:</p> <ul style="list-style-type: none"> - Restringir la actividad de un usuario (R14, Respuesta Activa de Protección y Respuesta Proactiva). - Terminar o reiniciar el servicio comprometido (R19 y R20, Respuesta Activa de Protección y Respuesta Activa de Recuperación). - Terminar o reiniciar el proceso sospechoso (R22 y R23, Respuesta Activa de Protección y Respuesta Activa de Recuperación). - Bloquear peticiones al sistema sospechosas (R24, Respuesta Activa de Protección y Respuesta Proactiva). - Agregar o borrar reglas de filtrado sobre un cortafuegos (R26 y R28, Respuesta Activa de Protección y Respuesta Proactiva). - Bloquear puertos o direcciones IP (R31, Respuesta Activa de Protección y Respuesta Proactiva). - Terminar conexión TCP establecida por el atacante (R34, Respuesta Activa de Protección). - Rastrear la conexión del atacante para recopilar información (R7, Respuesta Activa de Reacción).
---------------------------------	---	---	--

Fuente. (Mateos Lanchas, 2013), *adaptado y actualizado para propósitos de la investigación*



5.1.6 Inicio del Diseño y Construcción de Modulo de Proactividad

Tras haber revisado y haber encontrado evidencia que los Modelos Ocultos de Markov han sido usados para aplicar Proactividad en diferentes campos incluso en seguridad informática, como los trabajos realizados por **(Robayo Santana, 2009)** y **(Antolino Hernandez, 2011)**, se propone en este trabajo implementar el módulo de proactividad usando Modelos Ocultos de Markov con los cuales se utilizará mecanismos de Programación Evolutiva de tal manera que se logre que un HMM, creado aleatoriamente, evolucione para proporcionar un modelo confiable en la detección de anomalías. En el apartado 5.2 se detallara la arquitectura de la propuesta del módulo de proactividad, lo que se verá ahora es seleccionar de lo ya explicado en el Capítulo II los algoritmos a utilizar para lograr la evolución de los HMM.

5.1.7 Analizar y entender HMM y PE

Un análisis profundo de los Modelos Ocultos de Markov y Programación Evolutiva se ha mencionado en el Capítulo II. En este capítulo se explicara los retos a enfrentas con el uso de los HMM que clase de HMM se ha seleccionado y porque se usa la programación evolutiva.

Los HMM tratan con los problemas de:

- ✓ Reconocimiento de secuencias de observación,
- ✓ Secuencias probables de estados

✓ Parámetros de diseño óptimos de un HMM.

Para poder proporcionar respuesta a estos problemas se necesita conocer a priori la topología del modelo. Es decir, que para construir un HMM se necesita decidir cuantos estados tendrá, qué transiciones existirán entre sus estados y qué función de distribución se usará en los estados involucrados.

El principal problema que se encontró al momento de la construcción del módulo Proactivo es el de encontrar el mejor diseño o arquitectura de un HMM para realizar la tarea de detección de anomalías. El diseño se efectuará de manera automática, con sólo la secuencia de observación generando todos los parámetros necesarios y óptimos en la construcción del modelo.

El HMM modelará el perfil del comportamiento normal del contexto de red, basado en las métricas que se consideran en un contexto de red. Para entender lo que se entiende por comportamiento "normal", se puede observar un conjunto de variables del sistema y se puede decir que el sistema exhibe comportamiento normal o no normal, conocido como anómalo. A partir de las observaciones del comportamiento del sistema bajo condiciones normales (considerando que no se encuentra bajo ningún tipo de ataque), se captura la dinámica del sistema en un modelo probabilista. Este modelo probabilista nos indicará cuando el comportamiento del sistema se desvía considerablemente de su comportamiento normal. En estas condiciones decimos que el sistema presenta un comportamiento anómalo.



Una vez encontrado el mejor HMM que modele la secuencia de observación, es factible encontrar la solución al problema del reconocimiento de una secuencia de observación. El problema consiste en: dada una secuencia de Observaciones O , de una (o varias), variable(s), diseñar de manera automática un HMM λ que maximice la probabilidad de que la secuencia de observación haya sido generada por λ . Esto es

$$P(O | \lambda)$$

Con la finalidad de utilizarlo en la solución del problema de detección de ataques multipaso en red informática.

Un método utilizado para lograr encontrar los mejores parámetros de un HMM, son los GAs (Genetic Algorithms, Algoritmos Genéticos), y como se verá, el encontrar estos parámetros óptimos de los HMMs, es todo un reto. El adecuado inicio de valores de los parámetros de un HMM, repercute considerablemente en el desempeño del modelo. Determinar los mejores parámetros de un HMM es un problema de optimización difícil, debido a que los HMMs describen un doble proceso estocástico (Warrender, C., Forrest, S., & Pearlmutter, B., 1999).

Los HMMs tratan con estos problemas dentro de un marco probabilístico; ofrecen la ventaja de tener fuertes fundamentos estadísticos y ser computacionalmente eficientes. Pero tienen una desventaja: se necesita conocer de antemano la topología del modelo. Esto es, que para construir un HMM necesitamos decidir cuántos estados tendría y qué transiciones entre estados existirán.



Una vez que la estructura del modelo está diseñada, los parámetros de transición y emisión necesitarán ser estimados con los datos de entrenamiento.

El algoritmo de Baum-Welch (BW) se usa para entrenar los parámetros del modelo. Este algoritmo utiliza un método de Maximización de Expectativa, el cual dada una configuración inicial, ajusta los parámetros del modelo a una maximización local de probabilidad con los datos. El entrenamiento con el algoritmo de BW sufre del hecho de que encuentra un máximo local, y esto es sensible a la configuración inicial de los parámetros y el rendimiento del modelo. Para más información al respecto de los HMMs, véase (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989).

Como solución a esta problemática de la convergencia a un máximo local en el algoritmo de Baum-Welch, es la utilización de Algoritmos Genéticos (GAs). Los GAs utilizan una técnica de optimización global que puede ser usada en la optimización de los parámetros de un HMM.

5.1.8 Determinar y Aplicar Algoritmos para evolucionar HMM

Hasta aquí ya sabemos porque usaremos Programación Evolutiva para la optimización de los HMM, a continuación de da a conocer el aspecto interno que tendrá el módulo de proactividad, con este paso se está dando cumplimiento a los objetivo específicos “B” y “C”. El modulo esta implementado por dos sub módulos separados, uno se encargara de la evolución de los HMMs, entrenados con la secuencia de observación proporcionada para el entrenamiento, el otro sub modulo, una vez generado el HMM con EP, tiene la función de utilizar al HMM en el análisis de las secuencias de observación de prueba.

Existen trabajos de otros autores que han aplicado la computación evolutiva en el diseño y optimización de sistemas conexionistas. Por ejemplo (Jacob, C., 2001) presenta un esquema para evolucionar autómatas de estado finitos. (Antolino Hernandez, 2011) a través de su tesis contribuye indicando algoritmos evolutivos para evolucionar HMM los cuales se menciona a continuación:

Para poder efectuar un proceso evolutivo, el sistema comienza creando una población de individuos, representados por cromosomas, de manera aleatoria. El número de cromosomas se define como un parámetro al momento de ejecutar el sistema.

Posteriormente, cada cromosoma HMM de la población inicial generada aleatoriamente, se tiene que evaluar con respecto a la serie de tiempo dada y la función objetivo. Con estos dos elementos se

pueden calificar los HMMs, o en otras palabras, determinar su valor de aptitud.

Una vez que la población inicial ha sido evaluada, se toman los cromosomas mejor calificados y se les aplican los operadores de mutación evolutivos, como son: agregar o quitar algún estado, agregar o quitar una transición entre estados, mutar el valor de transición entre estados, mutar el valor del vector de medias y de la matriz de varianzas.

Una vez generados los cromosomas hijos, cromosomas mutados por los operadores evolutivos mencionados, se agregan a la población original, y son evaluados. Después, nuevamente se seleccionan los mejores cromosomas que pasan a la siguiente generación, donde se vuelve a repetir el ciclo del proceso evolutivo y selectivo de los mejores cromosomas de cada generación, hasta llegar al número de generación deseada.

Arquitectura de un HMM a evolucionar

Como se ha hecho mención, cada cromosoma de una población generada inicialmente, es en realidad una estructura completa de un HMM. Cada cromosoma se encuentra constituido por los siguientes genes: valor de Aptitud, Tamaño, Matriz de Transiciones entre estados, Matriz de Medias, Matriz de Covarianzas, y el vector Π .



La Tabla 1 muestra el cromosoma completo compuesto de los genes mencionados, y a continuación se describen las características particulares de cada gen.

Tabla 12. Arquitectura de un HMM o Cromosoma

Aptitud	Tamaño	Matriz de Transiciones	Matriz de Medias	Matriz de Covarianzas	Vector Pi
---------	--------	------------------------	------------------	-----------------------	-----------

Fuente. (Antolino Hernandez, 2011)

Aptitud. Representa el valor de aptitud del individuo, después de haber sido evaluado por la función objetivo.

Tamaño. Representa el número de estados que contiene el HMM.

Matriz de Transiciones. Contiene la matriz de probabilidad de transiciones entre los estados que constituyen al HMM.

Se muestra un esquema en la Tabla 2, donde $a_{i,j}$

$= P(q_{t+1} = S_j | q_t = S_i)$; con $1 \leq i, j \leq N$. Esto es, cada celda $a_{i,j}$ representa la probabilidad de transitar a S_j en un tiempo $t + 1$, dado que en el tiempo t se está en S_i . Con la propiedad de $\sum_{j=1}^N a_{i,j} = 1; \forall i$.

Tabla 13. Gen que representa la Matriz de Transiciones.

Estados	S1	S2	...	SN
S1	$a_{1,1}$	$a_{1,2}$...	$a_{1,N}$
S2	$a_{2,1}$	$a_{2,2}$...	$a_{2,N}$
.
S	$a_{N,1}$	$a_{N,2}$...	$a_{N,N}$

Fuente. (ANTOLINO HERNANDEZ, 2011)



Matriz de Medias. Esta matriz se encuentra constituida por los vectores de los valores de las medias de los estados. Cada media representa el valor promedio aritmético de las secuencias observadas de cada variable, con las cuales se entrenan los modelos. Es decir, esta matriz contiene la media de la distribución de la matriz B de cada estado.

La matriz B en cada estado representa la probabilidad de que se emita un símbolo en cada estado. Se puede observar este esquema en la Tabla 3.

Tabla 14. Gen que guarda la Matriz de Medias del Cromosoma.

Estado	Medias		
	X_1	...	X_d
S	$\mu_{1,1}$...	$\mu_{1,d}$
S	$\mu_{2,1}$...	$\mu_{2,d}$
.
SN	$\mu_{N,1}$...	$\mu_{N,d}$

Fuente. (Antolino Hernandez, 2011)

Matriz de Covarianzas. Los valores de la matriz son utilizados, al igual que la matriz de medias, como parámetros de entrada en la función de densidad de probabilidad (pdf). La función pdf que se utiliza es la distribución multigaussiana, que se encuentra definida en cada estado que constituyen al HMM. Las matrices de medias y covarianzas componen la matriz (B), la cual permite calcular la probabilidad de observar un símbolo en un estado determinado. Se puede observar la estructura de la matriz en la Tabla 4.

Tabla 15. Gen que mantiene la Matriz de Covarianzas para cada Estado

VARIABLES ALEATORIAS	X_1	X_2	...	X_d
-------------------------	-------	-------	-----	-------



X_1	$\sigma_{1,1}^2$			
X_2		$\sigma_{2,2}^2$		
.			..	
X_d				$\sigma_{d,d}^2$

Fuente. (Antolino Hernandez, 2011)

Vector Pi. Este último gen (π_i), ver Tabla 5, representa la probabilidad de que la primera observación haya sido generada en el estado S_i . Donde $1 \leq i \leq N$.

Tabla 16. Gen con el Vector Pi del Cromosoma

S_1	S_2	...	S_N
-------	-------	-----	-------

Fuente. (Antolino Hernandez, 2011)

Función objetivo

Para poder determinar el grado de aptitud que posee cada HMM, o cromosoma de la población, con respecto a la serie de tiempo dada, utilizamos la función objetivo descrito en la Ecuación

$$P(O | \lambda) = \sum_{i=1}^N \alpha_T(i)$$

Donde $\alpha_T(i)$ representa a la variable de avance α (Forward) (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989), la cual calcula la probabilidad total de la secuencia observada dado un HMM. Es decir, $P(O|\lambda)$ significa la



probabilidad que tiene el HMM de haber sido el modelo que generó la secuencia de observación. En otras palabras, significa que tan bueno es el modelo de HMM con respecto a la secuencia dada.

Los algoritmos evolutivos a implementar son mencionados y mostrados en pseudocódigo en el Capítulo II y han sido tomados de (Antolino Hernandez, 2011), cabe indicar que dichos algoritmos han sido implementados usando la aplicación Mathematica Versión 8.0 y en primera instancia se usara esta misma plataforma para entrenar los HMM y evaluar resultados de forma rápida, una vez comprobado esto se procederá a implementar dichos algoritmos en lenguaje C para lograr la implementación del módulo de proactividad y la unión al AIRS.

A continuación se detalla la lista de algoritmos usados en la programación del módulo de proactividad el detalle de los algoritmos esta mencionada en el apartado 2.3.7 del Capítulo II: Marco Teórico.

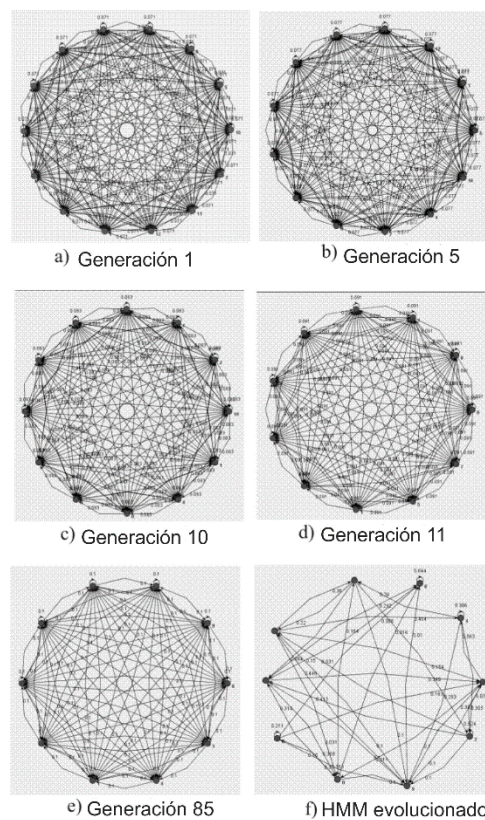
Los algoritmos usados son los siguientes:

- ✓ Algoritmo Evolution
- ✓ Algoritmo GenInitPop
- ✓ Algoritmo InitChrome
- ✓ Algoritmo Initialization
- ✓ Algoritmo Induction
- ✓ Algoritmo Forward
- ✓ Algoritmo FuncProbMult
- ✓ Algoritmo AddTransition
- ✓ Algoritmo DeleteTransition
- ✓ Algoritmo MutParameters
- ✓ Algoritmo Muttransitions
- ✓ AlgoritmoDeleteState
- ✓ Algoritmo AddState

5.1.9 Obtención de HMM optimizados

En la generación de los HMMs, el sistema hace evolucionar estos modelos usando EP y un conjunto de operadores particulares (como agregar-eliminar nodo, Agregar-Eliminar transición, etc.). Se probó la evolución de los modelos con diferentes números de generaciones y diferentes tamaños de población. La Figura 16 muestra los resultados de como la EP evoluciona un HMM en el sistema desarrollado en este trabajo de investigación. Cada sub-figura representa un paso en el proceso de creación y optimización de los HMMs, presentando al mejor HMM en cada generación.

Figura 15. Evolución de un HMM con EP



Fuente. Elaboración Propia



En la sub-figura a) de la Figura 16, se presenta la generación 1 del proceso evolutivo, donde se observa un HMM formado por 14 estados en su proceso evolutivo. La sub-figura b) muestra al HMM evolucionado con 13 estados hasta la generación 5. En la sub-figura c) se observa al HMM con 12 estados en la generación 10. En la sub-figura d) se muestra el HMM que ha evolucionado a sólo 11 estados en la generación 11. En la sub-figura e) se presenta al HMM con 10 estados, evolucionado hasta la generación 85. En la última sub-figura f) se presenta al HMM con 10 estados, y el proceso continúa hasta llegar al número de generación especificado. En este experimento se estableció en 120 generaciones. En la última generación, la EP ha alcanzado un número óptimo de estados y se concentran en refinar las probabilidades de transición.

5.1.10 Propuesta de arquitectura de AIRS Proactivo

En esta sección se propone la arquitectura de un sistema de respuesta automática frente a intrusiones proactivo.

5.1.10.1 Requisitos Funcionales y no funcionales de la arquitectura

Previa a la definición y especificación de la arquitectura del AIRS proactivo, se cree conveniente enumerar los requisitos del sistema, cuyo objetivo es describir el comportamiento externo del sistema que se va a desarrollar y sus restricciones operativas. Se distinguen:

- ✓ Requisitos funcionales: aquellos que definen las funciones del sistema o de sus componentes. Pueden ser frases muy generales

que definen acciones fundamentales que debe realizar el sistema, y cómo debe reaccionar ante situaciones particulares.

- ✓ Requisitos no funcionales: aquellos que especifican restricciones que afectan a los servicios o funciones del sistema, como por ejemplo, requisitos de rendimiento, seguridad, accesibilidad, usabilidad, estabilidad, interoperabilidad, escalabilidad, etc.
- ✓ Requisitos de entrada: aquellos que especifican las entradas soportadas por el sistema.
- ✓ Requisitos de salida: aquellos que definen la(s) salida(s) proporcionada(s) por el sistema.

Para cada una de las categorías (funcionales, no funcionales, entrada y salida), a modo de referencia se enumeran a continuación. Estas referencias son usadas para comentar por qué han sido propuestos, esto es, qué objetivo cumplen. Es importante destacar que mediante estos requisitos se listan las funcionalidades que el sistema debe cumplir.

Cabe mencionar que los requisitos están asociados al AIRS Proactivo como sistema global, es decir al conjunto de todos los módulos que componen la arquitectura propuesta.

5.1.10.2 Requisitos funcionales

Los requisitos funcionales del AIRS Proactivo se mencionan en la siguiente tabla:

Tabla 17. Requisitos No Funcionales del AIRS Proactivo

Referencia	Requisito
REQFUN01	El sistema responderá a ataques de red
REQFUN02	El sistema responderá a ataques de host
REQFUN03	El sistema prevendrá ataques futuros (Proactivo)
REQFUN04	El sistema será adaptable (tendrá en cuenta la efectividad o éxito de las respuestas en el pasado)
REQFUN05	El sistema será rápido en la emisión de respuestas
REQFUN06	El sistema medirá costo de respuesta (tendrá en cuenta el impacto del ataque)
REQFUN07	El sistema comprenderá la semántica de las alertas.
REQFUN08	El sistema tendrá en cuenta la relevancia del activo comprometido
REQFUN09	El sistema tendrá en cuenta el contexto asociado a cada escenario de intrusión
REQFUN10	El sistema considerará las alertas en función de la confianza o fiabilidad de los IDSs que la generan
REQFUN11	El sistema proporcionará un histórico de alertas y respuestas para su uso en escenarios colaborativos.

Fuente. (Mateos Lanchas, 2013), adaptada y actualizada para fines de la investigación por el autor

5.1.10.3 Requisitos No funcionales

Tabla 18. Requisitos No Funcionales del AIRS Proactivo

Referencia	Requisito
REQNOFUN01	El sistema debe ser desarrollado en JAVA casi en su totalidad
REQNOFUN02	Deberá ser configurable
REQNOFUN03	Deberá operar sobre sistema operativo Linux, dejando abierta la posibilidad de hacerlo portable.
REQNOFUN04	Debe ser escalable de tal manera que se pueda agregar nuevos módulos en trabajos futuros.
REQNOFUN05	Debe tener la capacidad de almacenar logs de operación

Fuente. (Mateos Lanchas, 2013), adaptada y actualizada para fines de la investigación por el autor



5.1.10.4 Requisitos de entrada

Tabla 19. Requisitos de entrada del AIRS Proactivo

Referencia	Requisito
REQENT01	El sistema aceptara entradas en diversos formatos sintácticos
REQENT02	Las entradas pueden incorporar contenidos que proporciones semántica

Fuente. (Mateos Lanchas, 2013), adaptada y actualizada para fines de la investigación por el autor

5.1.10.5 Requisitos de Salida

Tabla 20. Requisitos de salida del AIRS Proactivo

Referencia	Requisito
REQSAL01	El sistema podrá generar múltiples tipos de respuesta en base a un catalogo de respuestas
REQSAL02	Las respuestas serán reactivas (activas o pasivas) o proactivas
REQSAL03	El sistema generara registro históricos de ataques y respuestas

Fuente. (Mateos Lanchas, 2013), adaptada y actualizada para fines de la investigación por el autor

5.1.10.6 Diseño e Implementación de la arquitectura propuesta

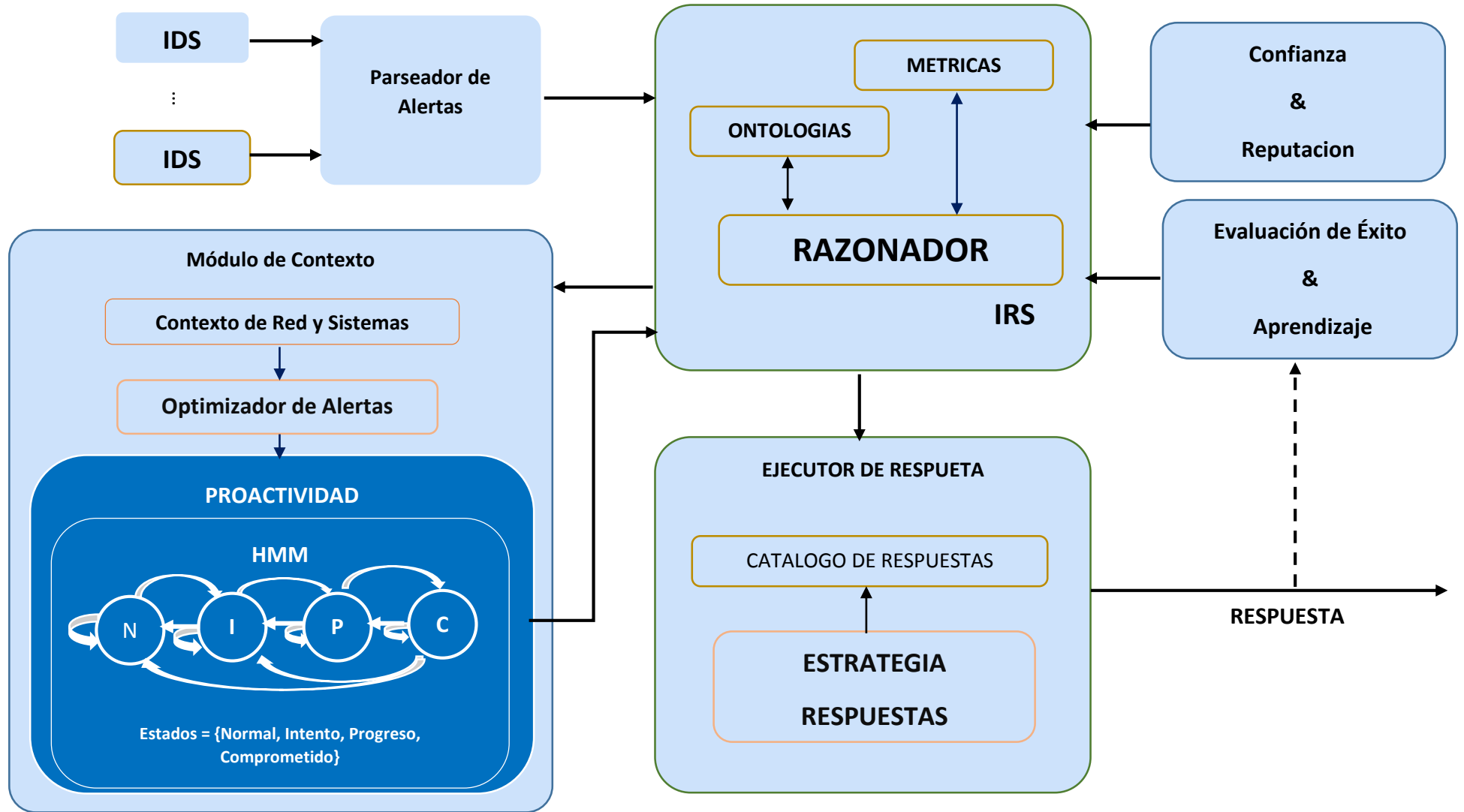
Una vez definidos los requisitos funcionales, de entrada y de salida del AIRS Proactivo, el objetivo de este punto es presentar los aspectos más relevantes del diseño de su arquitectura. En este apartado debo mencionar que se detallara en mayor proporción el módulo de proactividad ya que es el componente principal del desarrollo de este trabajo de tesis, de igual manera se hará para los módulos que se han desarrollado para completar la arquitectura del AIRS, respecto al resto de módulos como: Evaluador de éxito, catálogo de respuestas, contexto de red y de sistemas han sido



tomados de (Villagra, Martinez Perez, & Mateos Lanchas, 2013), cuyo código fuente está disponible en <http://reclamo.inf.um.es>. Se ha tomado solo los módulos que se han creído convenientes para la propuesta presentada.

La arquitectura que se propone en este trabajo cumple con los requisitos definidos previamente. En la siguiente figura 17, se observa que la arquitectura consta de 14 componentes, 1 componente externo (IDSs que proporcionan las entradas al sistema de respuesta) y 13 componentes principales de la arquitectura: Parseador de Alertas, Contexto de Red, Contexto de Sistemas, Optimización de Alertas, Proactividad, Evaluación de Éxito y Aprendizaje, Ontología de Respuesta a Intrusiones, Métricas (Políticas), Razonador, Catalogo de Acciones de Respuesta, Ejecutor de acciones de respuesta, Confianza y Reputación e interfaz de administración. Cada módulo funciona de manera independiente y para su integración basta con conocer las interfaces de entrada y salida respectivas. Se puede observar en color celeste los módulos utilizados del AIRS RECLAMO y en color azul los módulos agregados por este proyecto de investigación.

Figura 16. Arquitectura Propuesta del AIRS Proactivo



Fuente. Elaboración Propia



La arquitectura propuesta es una arquitectura modular, El objetivo de esta arquitectura es inferir la respuesta óptima de entre un conjunto de respuestas disponibles en la organización, frente a intrusiones detectadas en cualquier lugar de la misma. A rasgos generales y con el objetivo de dar una visión global del funcionamiento del sistema, el proceso seguido es el siguiente: El administrador del sistema define un conjunto de respuestas especificando la acción que desempeña, su coste, severidad, impacto, eficacia etc. Esta información acerca de las respuestas predefinidas se incluye en la ontología de respuesta a intrusiones. Cuando un IDS de los existentes en la red detecta una intrusión, envía una alerta de intrusión al AIRS, evento que desencadena el proceso de inferencia. En dicho proceso, el Razonador ejecuta las Políticas o reglas que especifican el comportamiento del sistema e infiere una de las respuestas previamente definidas, teniendo en cuenta para ello la información contenida en la ontología de respuesta a intrusiones relativa a la alerta de intrusión recibida, al contexto de la red y los sistemas en el momento de la intrusión, al activo comprometido y a las acciones de respuesta. Los resultados inferidos son enviados al Ejecutor de Acciones de Respuesta, que se encarga de ejecutarla. Finalmente, el Evaluador del éxito calcula si la acción desplegada ha conseguido mitigar la intrusión o si, por el contrario, no ha tenido ningún efecto sobre ella, realimentando esta información en la ontología de respuesta a intrusiones para futuras ejecuciones.

A continuación se describe cada componente presente en la arquitectura y la función que desempeña en el proceso de inferencia.

IDSs

Los IDSs son sistemas de detección de intrusiones que se encuentran distribuidos en la red de la organización, cuyo objetivo es detectar comportamiento anómalo y enviar el informe de intrusión correspondiente al AIRS Proactivo, concretamente al paseador de Alertas. Este informe contiene parámetros relativos al ataque detectado, entre los que se encuentran el tipo de intrusión o la dirección IP a la que va dirigida el ataque, parámetros necesarios para el correcto funcionamiento del sistema de respuesta. La arquitectura soporta cualquier tipo de IDS, pero en la validación de la misma se han usado NIDSs y HIDSs. Detalles acerca de la integración de distintos IDSs específicos con la arquitectura así como del formato de alerta de intrusión utilizado, se proporcionan en el apartado de validación de la propuesta.

Cabe también indicar que los IDSs son componentes externos a la arquitectura del sistema de respuesta que pueden estar localizados en hosts distintos al host donde reside el AIRS o en el mismo host. A pesar de ser externos, son una pieza fundamental para el funcionamiento del sistema de respuesta, ya que su función es desencadenar el proceso de inferencia de la respuesta cada vez que detectan comportamiento anómalo en la red de la organización.

Desde el punto de vista de la implementación del prototipo del AIRS Proactivo, la arquitectura propuesta soporta dos IDSs: Snort como NIDS y OSSEC como HIDS.

En ambos casos, el informe de intrusión se genera en formato Syslog, y en el caso de Snort este informe tiene la siguiente estructura:

Figura 17. Estructura de informe de alertas de IDS Snort

```
<priority> month day time hostname process[pid]: [x:y:z] message  
[Classification: classification] {protocol} source -> destination
```

Fuente. (Mateos Lanchas, 2013)

Por otra parte, con el fin de dotar de transparencia al sistema y dificultar la detección de la existencia de NIDSs en la red por parte de los atacantes, estos funcionan en modo promiscuo escuchando en una interfaz de red el tráfico que desean analizar, y se utiliza un mecanismo denominado port mirroring, que consiste en que el switch o router actúa como un espejo, duplicando todo el tráfico entrante o saliente de una determinada interfaz y enviando la copia a la interfaz donde el NIDS (Snort en la arquitectura propuesta) está escuchando. Hay varias técnicas que permiten implementar el mecanismo de port mirroring, bien configurando de forma conveniente los switches y routers, siguiendo las especificaciones proporcionadas por sus desarrolladores (por ejemplo: mediante los comandos monitor session 1 source interface interface1 – monitor sesión 1 destination interface interface2, en el caso del Cisco Catalyst 3750; port monitor interface1 en el Cisco Catalyst 3500XL; o set mirror port source interface1 destination interface2 en el caso de Juniper/Netscreen Firewall), o bien utilizando otras técnicas como daemonlogger. Esta última

opción es la técnica utilizada en la implementación del AIRS Proactivo. Daemonlogger actúa como sniffer sobre una interfaz y envía un duplicado de todo el tráfico recibido hacia otra interfaz, en este caso en la que está escuchando el NIDS Snort. Para ello, tan sólo es necesario indicar la interfaz cuyo tráfico va a ser duplicado y la interfaz a la que va a ser enviado, para lo que es necesario ejecutar los siguientes comandos:

Figura 18. Comando a ejecutar para habilitar técnica Daemonlogger

```
daemonlogger -i NetATT -o IDSNet1
```

Fuente. (Mateos Lanchas, 2013)

Con el uso del mecanismo port mirroring utilizando daemonlogger se consiguen dos objetivos:

- ✓ Ocultar en cierta medida la existencia de NIDSs.
- ✓ No contaminar o modificar el tráfico original enviado desde el atacante a la máquina víctima.

En la Figura 20 se observa como el daemonlogger instalado sobre el host R-FW copia los paquetes que llegan por la interfaz NetATT y los envía hacia la interfaz IDSNet1, subred en la que se encuentra conectado el IDS Snort. Como se observa, esto se hace de una forma transparente al atacante, puesto que los paquetes además son enviados hacia su destino.

En el caso de los HIDSs no es necesario utilizar este mecanismo puesto que analizan todo el tráfico entrante y saliente al host en el que residen.

En ambos casos, los IDSs monitorizan el tráfico ya sea a nivel de red o de host, y ante la detección de una intrusión generan una alerta en formato

Syslog, como se ha indicado previamente. Esto requiere la configuración de los archivos de configuración de Snort y OSSEC:

En el caso de Snort, para especificar Syslog como formato de salida, es necesario añadir la siguiente línea en el fichero de configuración snort.conf: output alert_syslog: LOG_AUTH LOG_ALERT.

En el caso de OSSEC, es necesario añadir lo siguiente al fichero de configuración ossec.conf:

La dirección del servidor syslogd:

```
<syslog_output>
    <server>10.0.1.1</server>
    <port>514</port>
</syslog_output>
```

La habilitación del cliente syslog de OSSEC:

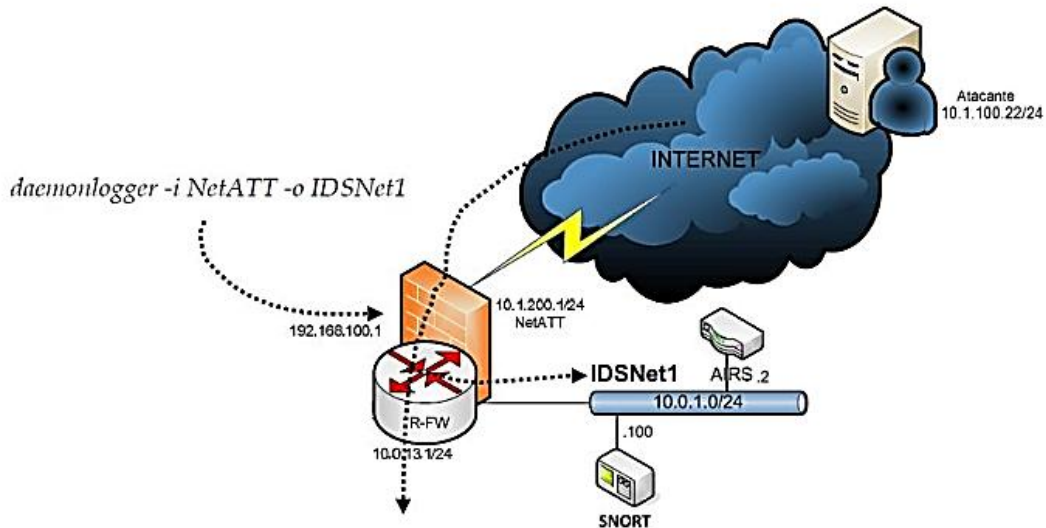
```
/var/ossec/bin/ossec-control enable client-syslog
```

Cada vez que uno de los IDSs presentes en la arquitectura de la institución detecta una intrusión, genera una alerta que envía hacia el Parseador de Alertas, cuya configuración desde el punto de vista de la implementación se especifica en el siguiente apartado.

La configuración de los IDSs va más allá de lo explicado en este apartado, es decir, además de lo especificado es necesario definir las reglas de detección y filtrado, habilitar los preprocesadores correspondientes, definir

los archivos o sistemas a monitorizar, etc., dependiendo estas tareas de las necesidades específicas de cada organización.

Figura 19. Uso de Daemonlogger como técnica Port Mirroring



Fuente. (Guaman Loachamin, 2013)

Parseador de Alertas

Este componente de la arquitectura recibe las alertas de intrusión procedentes de diferentes IDSs o fuentes de alertas con diferentes formatos y sintaxis, y mapea los conceptos incluidos en estas alertas de intrusión a los conceptos equivalentes definidos en la ontología. De esta manera, conceptos semánticamente equivalentes pero sintácticamente diferentes se mapean a un mismo concepto en la ontología, en el caso en que el AIRS Proactivo reciba dos informes de intrusión relativos a la misma intrusión de diferentes fuentes con sintaxis diferente.

Este componente se encarga, pues, de recibir y mapear cualquier alerta de intrusión enviada por cualquier fuente de detección con formato

diferente a un formato de alerta común, comprensible por el razonador del AIRS Proactivo propuesto. Entre los requisitos del parseador de alertas se resalta que debe ser capaz de recibir alertas de intrusión procedentes de distintos tipos de IDSs, como por ejemplo de NIDSs y de HIDSs.

Desde el punto de vista de su implementación, este componente está diseñado para funcionar como un servidor de alertas Syslog, que escucha a través del puerto UDP 512. De forma breve el proceso seguido es el siguiente:

- ✓ Cuando se inicia el AIRS Proactivo, este componente es iniciado. Desde ese momento, permanece a la espera de recibir alertas de intrusión escuchando en el puerto 512.
- ✓ Una vez que cualquiera de los IDSs del sistema detecta un ataque, genera y envía el informe de intrusión correspondiente. Por defecto, esta alerta en formato syslog se envían al puerto 514 (puerto por defecto del servicio syslogd) del host donde reside el núcleo del AIRS Proactivo.
- ✓ -El siguiente paso es redirigir todo mensaje recibido en el puerto 514 hacia el puerto 512, donde está escuchando el módulo de Parseador de Alertas del AIRS Proactivo. Para ello, basta con añadir la siguiente regla al cortafuegos IPTables:

Figura 20. Regla de Cortafuegos para redirigir mensaje del puerto 514 hacia el puerto 512

```
iptables -A PREROUTING -t nat -p udp -dport 514 -j DNAT -to-destination
IP_AIRS:512
```

Fuente. (Guaman Loachamin, 2013)



- ✓ A continuación, el Receptor de Alertas mapea la alerta recibida a un formato común de alerta utilizado en la arquitectura del AIRS Proactivo
- ✓ Una vez que la alerta ha sido mapeada, comienza el proceso de inferencia de la respuesta óptima para lo que este componente invoca al Razonador del AIRS pasándole como parámetro la alerta de intrusión mapeada, la máscara de la subred en la que se encuentra el AIRS y una variable que representa el instante actual. Para ello, el Receptor de Alertas ejecuta lo siguiente:

```
Runnable nuevaAlerta = new OntAIR(alertarecibida, netMask,
currentDate);
exec.execute(nuevaAlerta);
```

Contexto de Red y Contexto de Sistemas

Estos componentes de la arquitectura tienen como función capturar y analizar información del contexto en tiempo real, cada vez que una nueva intrusión ha sido detectada y el AIRS ha recibido una alerta de intrusión. La diferencia entre ambos componentes es el tipo de información que capturan.

El Contexto de Red captura el tráfico en la red de la organización y para cada paquete analiza diferentes parámetros tales como fecha, tipo de protocolo, dirección IP origen, puerto origen, dirección IP destino, puerto destino, y número de bytes recibidos y transmitidos. Una vez obtenidos estos parámetros en tiempo de intrusión, calcula un parámetro denominado Anomalía de Red que indica el grado de anomalía o anormalidad de la red respecto a un estado de funcionamiento habitual.

Para ello, compara los indicadores obtenidos en el momento del ataque con un perfil o patrón del tráfico de la red previamente generado.

Por su parte, el Contexto de Sistemas compara el valor de varios indicadores del sistema comprometido antes y después de la intrusión, como pueden ser el número de procesos activos en el sistema, el uso de la CPU, el número de procesos zombies, el espacio libre en disco, el número de usuarios logueados, el estado del sistema, el número de intentos fallidos de conexión por SSH o la latencia del sistema. Al igual que en el contexto de red, es necesario generar un perfil o patrón del contexto relativo a cada sistema en tiempo normal de operación, previamente. En el momento de intrusión, este componente se encarga de comparar el valor de cada indicador con su valor en el perfil y obtener el grado de anomalía existente.

Ambos componentes son invocados por el Razonador del sistema, cuando este ha recibido una alerta de intrusión mapeada al formato común, y ha comprobado que es necesario inferir una acción de respuesta frente a la intrusión detectada. El Razonador, antes de invocar a los módulos de contexto, comprueba que la alerta recibida no es una alerta repetida ni es continuación de un ataque existente, para lo que utiliza métricas de seguridad y reglas o políticas especificadas mediante SWRL. Una de las ventajas de utilizar ontologías y lenguajes de definición de comportamiento formales, radica en que permiten solventar el problema de la coherencia semántica que poseen otros AIRSs estudiados, de tal forma que alertas recibidas por el AIRS con diferentes sintaxis y formatos

pero referidas a la misma intrusión, sean consideradas como una única alerta.

Contexto de Sistemas

El contexto de sistemas está formado por tres componentes principales: un sistema de monitorización, un módulo de procesado de la información y la base de datos de perfiles de parámetros de los sistemas.

Sistema de monitorización

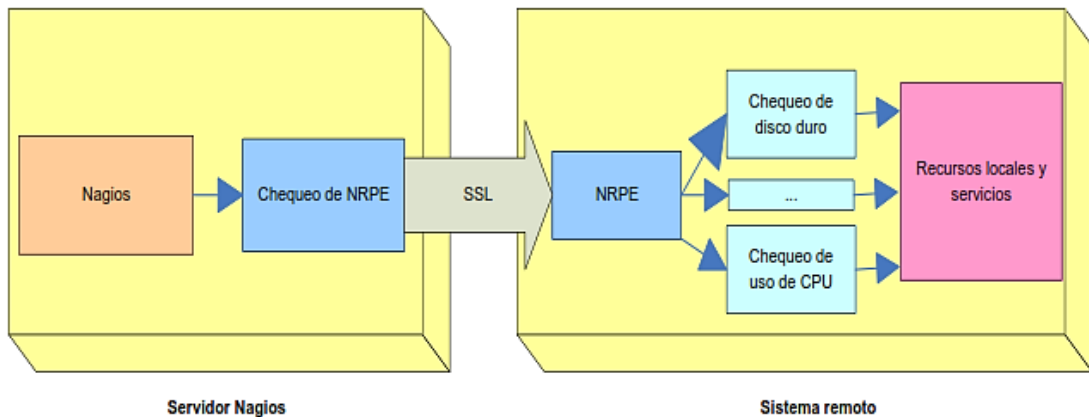
El sistema de monitorización a utilizar, en la implementación del prototipo del AIRS Proactivo es Nagios 7, en su versión 3.0, desarrollado bajo licencia GNU General Public License Version 2.

Nagios es un sistema de monitorización cuya arquitectura es del tipo cliente-servidor siendo su funcionamiento el típico de este tipo de sistemas, donde el servidor recupera la información de los distintos sistemas clientes para procesarla y generar estadísticos y perfiles, y los clientes son aquellos sistemas que se desean monitorizar. Para monitorizar los sistemas es necesaria la instalación de agentes que permitan recoger la información del dispositivo y enviarla al servidor cuando éste la solicite. Cabe destacar que los agentes que utiliza Nagios pueden funcionar en prácticamente cualquier sistema operativo y en muchos tipos de terminales, lo que aporta una gran flexibilidad a este módulo de

contexto. Para el correcto funcionamiento del sistema de monitorización, ha sido necesario seguir una serie de pasos:

- ✓ Instalación del servidor Nagios en el mismo host en el que reside el AIRS Proactivo. Este servidor se compone de una consola y de un servidor web que permite visualizar el estado de los equipos.
- ✓ Instalación de los agentes en los distintos terminales o hosts cuya información de contexto se quiera monitorizar, ya que sin los agentes únicamente se podría acceder a un tipo de información muy básica como la latencia o si el host está caído, interesante pero insuficiente para el propósito perseguido. Además, sería imposible obtener información de por ejemplo la carga de CPU o el número de usuarios. Existen diferentes agentes disponibles para instalar para todos los sistemas operativos. Como se indica en los requisitos no funcionales del AIRS Proactivo, en concreto en el REQNF04, “ El AIRS debe operar sobre el S.O. Linux y dejar abierta la posibilidad de extenderlos a otros S.O.”, por lo que el agente instalado ha sido el NRPE. NRPE es un add-on diseñado para permitir que se ejecuten plugins de Nagios en hosts remotos de forma segura. El esquema general que utiliza el servidor nagios y el cliente NRPE es el mostrado en la siguiente figura:

Figura 21. Esquema de funcionamiento del Contexto de Sistemas



Fuente. (Mateos Lanchas, 2013)

Siguiendo este funcionamiento el servidor Nagios es capaz de enviar peticiones de chequeo a los terminales remotos, que reconocerán las solicitudes según los procedimientos definidos en su agente NRPE. El protocolo SSL proporciona servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente con un algoritmo de cifrado simétrico, por lo que se asegura que la comunicación sea segura y los métodos sólo sean accesibles por el servidor Nagios instalado en el host donde reside el AIRS Proactivo.

- ✓ Una vez instalados los agentes de usuario hay que darlos de alta y configurarlos en el servidor, de la forma apropiada, de forma que sea posible monitorizar los 8 parámetros de sistemas especificados.

Tras finalizar los tres pasos anteriores, se tiene un sistema de monitorización funcionando como un sistema de monitorización estándar, pero no es suficiente para el módulo de contexto utilizado



en la arquitectura del AIRS Proactivo, ya que en este caso es necesario realizar chequeos inmediatos, y leer la información capturada automáticamente, durante la fase de detección de anomalías. Para ello, es necesario configurar de forma adecuada el archivo del servidor Nagios, `nagios.cfg`, de forma que sea posible obtener la información de monitorización en un instante específico (Nagios por defecto sólo realiza chequeos pasivos cada cierto tiempo), y devolver automáticamente dicha información al módulo de procesado encargado de analizarla. Para ello, es necesario habilitar la opción de que Nagios guarde la información en una serie de ficheros instantáneamente en el momento de la captura. En este caso concreto, se definen dos nuevos ficheros: Host Performance Data y Service Performance Data, cuyo contenido así como su formato de visualización también es necesario especificar en el archivo de configuración de Nagios. En el prototipo implementado el formato utilizado para dichos ficheros ha sido:

- **`service_perfdata_file_template=$HOSTNAME$!t$SERVICEDESC$!t$SERVICEOUTPUT$!t$SERVICEPERFDATA$`**
- **`host_perfdata_file_template=$HOSTNAME$!t$HOSTOUTPUT$!t$HOSTPERFDATA$`**

De esta forma, el módulo de contexto de sistemas podrá solicitar el valor de cada uno de los 8 parámetros configurados en un instante concreto, sin más que la ejecución de un script de Linux programado expresamente que ejecuta los comandos adecuados para realizar un chequeo inmediato de los ocho servicios de los



sistemas que queremos controlar. El módulo de Optimización de Alertas es el responsable de ejecutar dicho script.

Base de datos

Componente de la arquitectura del módulo de contexto de sistemas cuyo objetivo es almacenar la información relativa a los parámetros de los sistemas monitorizados durante la fase de aprendizaje, y proporcionar dicho perfil durante la fase de detección de anomalías.

En el prototipo desarrollado se utiliza MySQL para la creación de la base de datos, denominada SystemContextDB. Como se observa a continuación, está compuesta por la tabla STATUS, que a su vez se compone de una clave primaria autonumérica, el día de la semana, el momento dentro del día, el nombre del sistema, el estado de funcionamiento, el número de usuarios, la carga de la CPU, el número de procesos en ejecución, el espacio libre en el disco duro, la latencia, el número de procesos zombies y el número de conexiones SSH fallidas. El día de la semana y el momento dentro del día han sido incluidos como atributos para poder realizar las consultas teniendo en cuenta el factor temporal de una forma sencilla y eficiente.

```
CREATE DATABASE SystemContextDB;
CREATE TABLE IF NOT EXISTS STATUS (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    day_week INT,
    time LONG,
    name VARCHAR(32) NOT NULL DEFAULT "",
    state VARCHAR(32) NOT NULL DEFAULT "",
    users INT,
    CPU FLOAT,
    processes INT,
    free_space LONG,
    latency FLOAT,
    zombies INT,
    sshFailed INT
);
```

Durante la fase de aprendizaje el módulo de procesado de la información es el encargado de insertar nueva información en la base de datos, para lo que hace uso de la directiva INSERT.

A modo de ejemplo, la siguiente figura muestra el aspecto de la base de datos una vez comenzada la fase de aprendizaje, donde observan los distintos parámetros especificados que forman parte del perfil de actividad almacenado.

Figura 22. Módulo de Contexto de Sistemas. SystemContextDB

id	day_week	time	name	state	users	CPU	processes	free_space	latency	zombies
119	3	1125	remotehost	working	2	0	125	6536	0.72	0
120	3	1126	remotehost	working	2	0	119	6536	0.23	1
121	3	1127	remotehost	working	2	0	128	6536	0.32	0
122	3	1128	remotehost	working	2	0	132	6536	0.18	0
123	3	1129	remotehost	working	2	0	119	6536	0.17	0
124	4	668	remotehost	working	2	0.3	131	6536	0.27	0
125	4	669	remotehost	working	2	0.1	129	6536	0.51	0
126	4	671	remotehost	working	2	0.03	123	6536	0.17	0
127	4	672	remotehost	working	2	0.01	124	6536	4.86	1
128	4	673	remotehost	working	2	0	123	6536	0.84	0
129	4	674	remotehost	working	2	0	123	6536	0.22	1
130	4	675	remotehost	working	2	0	125	6536	0.19	0
131	4	676	remotehost	working	2	0	119	6536	0.27	0
132	4	677	remotehost	working	2	0	120	6536	0.43	0
133	4	678	remotehost	working	2	0	119	6536	0.24	0
134	4	680	remotehost	working	2	0	124	6536	0.22	0
135	4	733	remotehost	working	2	0	123	6536	0.22	0
136	4	734	remotehost	working	2	0	129	6536	0.21	0
137	4	735	remotehost	working	2	0	125	6536	11.58	0
138	4	736	remotehost	working	2	0	124	6536	0.2	0
139	4	741	remotehost	working	2	1.83	120	6062	75.03	0
140	4	742	remotehost	working	2	2.86	122	4911	32.58	0
141	4	743	remotehost	working	2	3.08	134	3789	68.3	0

Fuente. Captura de Pantalla Gestor MySQL

Por otra parte, durante la fase de detección de anomalías, el módulo de optimización de alertas consulta la base de datos para la obtención de información acerca del perfil del sistema comprometido, para lo que ejecuta el comando SELECT. Como resultado de esta consulta, el módulo de optimización de alertas recibe aquellas filas de la base de datos cuyo sistema se corresponda con el sistema comprometido, y la franja horaria sea equivalente a la franja horaria del momento de detección de la intrusión.



Contexto de Red

El contexto de red también está formado por tres componentes principales: un capturador de tráfico, un módulo de procesado de la información y la base de datos que contiene el perfil de actividad de la red monitorizada.

Capturador de tráfico

Tras un análisis realizado, en el desarrollo del módulo de contexto de red implementado como parte de la validación de la arquitectura del AIRS Proactivo, se utiliza SANCP como capturador de tráfico. Una descripción más detallada de este sistema se proporciona más adelante en este capítulo.

Base de datos

Componente de la arquitectura del módulo de contexto de red cuyo objetivo es almacenar la información relativa al tráfico habitual de la red durante la fase de aprendizaje, y proporcionar dicho perfil de tráfico durante la fase de detección de anomalías.

En el prototipo desarrollado se utiliza MySQL para la creación de la base de datos, denominada NetContextDB. Como se observa a continuación, esta base de datos contiene únicamente una tabla denominada CONNECTION, donde se almacena la información de las distintas

conexiones en la fase de aprendizaje. Los distintos atributos de la tabla CONNECTION corresponden a: clave de identificación autonumérica, fecha, día de la semana, momento del día, duración de la conexión, protocolo de Ethernet, protocolo de IP, puerto de origen, puerto de destino, IP de origen, IP de destino, número de paquetes de origen, número de paquetes de destino, número de bytes recibidos sin contar cabeceras, número de bytes enviados sin contar cabeceras, número de bytes totales contando cabeceras. El día de la semana y el momento dentro del día han sido incluidos como atributos para poder realizar las consultas teniendo en cuenta el factor temporal de una forma sencilla y eficiente.

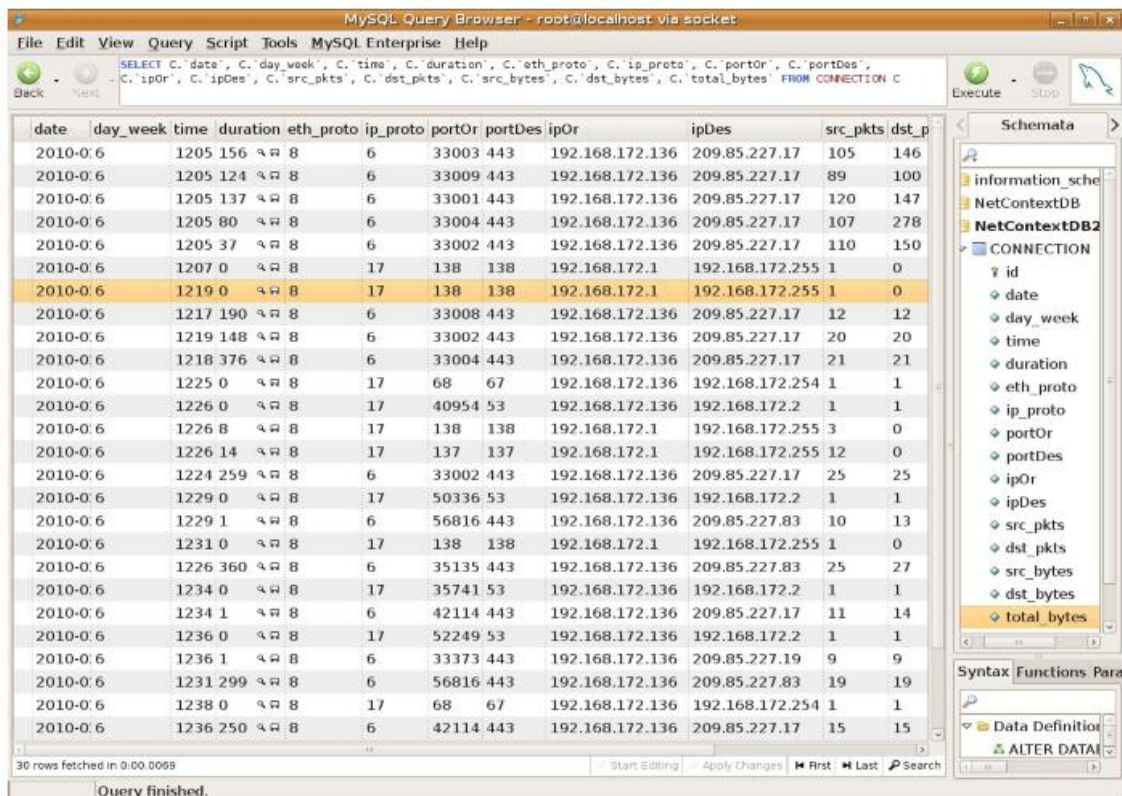
```
CREATE DATABASE NetContextDB;
CREATE TABLE IF NOT EXISTS CONNECTION (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    date VARCHAR(32) NOT NULL DEFAULT "",
    day_week INT,
    time INT,
    duration LONG,
    eth_proto INT,
    ip_proto INT,
    portOr INT,
    portDes INT,
    ipOr VARCHAR(32) NOT NULL DEFAULT "",
    ipDes VARCHAR(32) NOT NULL DEFAULT "",
    src_pkts INT,
    dst_pkts INT,
    src_bytes INT,
    dst_bytes INT,
    total_bytes INT
);
```

Durante la fase de aprendizaje el módulo de procesado de la información es el encargado de insertar nueva información en la base de datos, de la forma mostrada a continuación:

```
insert into CONNECTION (date, day_week, time, duration, eth_proto, ip_proto, portOr, portDes, ipOr, ipDes, src_pkts, dst_pkts, src_bytes, dst_bytes, total_bytes) VALUES ('2010-02-26 20:05:08',6,1205, 156,8, 6,33003,443, '192.168.172.136', '209.85.227.17', 105, 146, 14177, 145988, 173840);
```

A modo de ejemplo, la siguiente figura muestra el aspecto de la base de datos una vez comenzada la fase de aprendizaje, donde observan los distintos parámetros especificados que forman parte del perfil de tráfico almacenado, información obtenida de los logs de SANCP.

Figura 23. Módulo de Contexto de Red. NetContextDB



Fuente. Captura de Pantalla Gestor MySQL



Por otra parte, durante la fase de detección de anomalías, el módulo de procesado de la información consulta la base de datos para la obtención de información acerca del perfil de tráfico de la red, ejecutando un comando de la siguiente forma:

```
String select = "SELECT C.`date`, C.`day_week`, C.`time`, C.`duration`, C.`eth_proto`, C.`ip_proto`, C.`portOr`, C.`portDes`, C.`ipOr`, C.`ipDes`, C.`src_pkts`, C.`dst_pkts`, C.`src_bytes`, C.`dst_bytes`, C.`total_bytes` FROM CONNECTION C where day_week=" + day1 + " AND time >= " + time1 + " AND time <= " + time2;
```

En esta consulta se observa que aparte de solicitar todos los atributos (a excepción de la clave de identificación primaria que no es interesante para este propósito) se restringen las conexiones buscadas a las que se hubiesen obtenido en el mismo día de la semana que cuando se está produciendo la posible intrusión y en el mismo momento del día. De esta forma nos aseguramos que el perfil que se generará está compuesto por conexiones que son relevantes a la hora de realizar la correlación y el grado de anomalía se ajuste al máximo a la realidad de la situación de nuestra red.



Optimizador de Alertas

Existen 2 componentes bien diferenciados para el optimizador de alertas `systemContext` para el contexto de sistemas y `netContext` para el contexto de red.

Para el primero el objetivo del componente es la correlación de la información recibida del sistema de monitorización en tiempo de detección con la información almacenada en la base de datos durante la fase de aprendizaje. Desde el punto de vista de la implementación del mismo, este módulo es un programa Java que se compone de un paquete principal denominado `systemContext`, que engloba todo el módulo y una serie de paquetes secundarios que realizan una serie de subfunciones para obtener el grado de anomalía del sistema atacado. Los subpaquetes más importantes son:

- ✓ `SystemCorrelator`: la función de este paquete es la obtención del grado de anomalía en los parámetros del sistema atacado, para lo cual utilizará el perfil generado a partir de los datos extraídos de la base de datos y del sistema de monitorización directamente en el momento del ataque. El resultado se obtiene mediante la comparación de estos datos utilizando un algoritmo que finalmente devolverá un objeto que contendrá los grados de anomalía de cada uno de los parámetros del sistema monitorizados.
- ✓ `InfoLoader`: es el paquete encargado de obtener la información del sistema de monitorización.

Además tiene las funciones necesarias para realizar el proceso de aprendizaje mediante la carga periódica de los parámetros de los sistemas. Este paquete utiliza el paquete DAO para la inserción de la información extraída en la fase de aprendizaje en la base de datos.

- ✓ DAO: es el paquete que realiza las operaciones con la base de datos. Al haber utilizado MySQL, el driver que se utiliza es el compatible con éste gestor. Este paquete contiene las funciones tanto de inserción de nueva información en la base de datos como de extracción, para la generación de perfiles con los que comparar el estado del sistema concreto en el momento del ataque.
- ✓ Util: paquete que contiene una serie de clases que apoyan al resto de paquetes pero por su tipo de funcionalidad de carácter más generalista no pertenecen directamente a ninguno de los anteriores.

Para realizar la correlación entre el estado normal del sistema y el estado en el momento que se detecta un posible ataque, es necesario generar un perfil que contenga el comportamiento habitual de dicho sistema en un momento de la semana similar al momento en el que se ha producido la intrusión, ya que parámetros como por ejemplo, la carga de la CPU, pueden variar en gran medida desde la mañana hasta media noche según el día de la semana que sea. Dicho perfil se genera en la fase de aprendizaje del módulo de contexto. En tiempo de intrusión, el método correspondiente del módulo de procesado de la información obtiene por un lado el perfil del comportamiento del sistema comprometido almacenado anteriormente en la base de datos, y por otro lado el estado



de cada uno de los parámetros del sistema en momento de detección. A partir de los datos obtenidos de la base de datos crea un objeto que contiene las medias y desviaciones cuadráticas de cada parámetro monitorizado, tras lo cual compara parámetro a parámetro la información contenida en el objeto creado con el estado actual del sistema, para lo que hace uso de una distribución de probabilidad. Esta función de distribución se realiza mediante un estadístico que tiene en cuenta no sólo la media, sino también las diferentes varianzas, ya que no representa el mismo grado de anomalía en el disco duro por ejemplo, una variación del 10 % en memoria libre, en un servidor de descarga y subida de archivos, que en un servidor Web cuyo contenido sea prácticamente constante, donde ésta variación sería mucho más sospechosa. La probabilidad que se obtiene de esta distribución acumulativa proporciona la confianza del dato actual respecto a su comportamiento típico, luego la anomalía será su complementario. Los números se han ajustado en función de los porcentajes de confianza para que el resultado sea un número del cero al diez para cada parámetro, representando dicho valor la anomalía de ese parámetro en el sistema concreto.

Como resultado, el módulo de procesado de la información proporciona un objeto de tipo SystemAnomaly que contiene el grado de anomalía existente en cada uno de los ocho parámetros monitorizados.

Para el caso de netContext el objetivo de este componente es la correlación de la información recibida del capturador de tráfico en tiempo de detección con la información almacenada en la base de datos durante



la fase de aprendizaje relativa al tráfico habitual en la red o subred bajo monitorización. Desde el punto de vista de la implementación del mismo, este módulo es un programa Java que se compone de un paquete principal llamado netContext que está compuesto por un interfaz y un conjunto de clases de carácter general, además de por una serie de subpaquetes. A continuación se detalla la función de cada uno de los subpaquetes de forma general:

- ✓ correlatorEngine: paquete encargado de realizar la correlación entre el perfil que genera con la información que obtiene de DAO y la Snapshot (información instantánea capturada). Como resultado devuelve un número que contiene el grado de anomalía de la red.
- ✓ DAO: paquete que se encarga del acceso a una base de datos que contiene toda la información capturada durante la fase de aprendizaje comentada. Tendrá funciones tanto de consulta como de inserción.
- ✓ Sancp: su principal función es conseguir generar la Snapshot comentada mediante la puesta en funcionamiento del programa SANCP en modo realtime y realizar el procesado de dicha información.
- ✓ Converter: el objetivo de este paquete es realizar la operación de parseo de una línea del log de SANCP y procesarla. Para ello hay que destacar que el formato indicado en sancp.conf debe coincidir con el utilizado en dicho paquete.
- ✓ infoLoader: este paquete es el encargado de procesar un archivo del log completo, por ello también hará uso del paquete converter.

- ✓ util: este paquete ha sido desarrollado para aportar una serie de funciones que ayuden y simplifiquen el resto de paquetes, sin pertenecer de forma natural a ninguno de ellos.

Como resultado de este módulo se obtiene un parámetro que se representa con un número del 0 al 10, y se corresponden con la anomalía detectadas por la función de correlación sobre el tráfico de red. Además éste será el valor que devuelva este módulo como respuesta a la petición del Razonador.

El proceso de correlación es bastante complejo, pero se intentará dar una idea del procedimiento seguido para llegar al resultado. En primer lugar se genera un objeto de tipo Profile, a partir del acceso a la información de la base de datos mediante el uso de la interfaz correspondiente, que devuelve las conexiones que servirán para generar el perfil. Las conexiones que son seleccionadas de la base de datos dependen fundamentalmente del momento del día en el que se produjeron y del día de la semana, permitiendo así comparar periodos que teóricamente deberían tener unas características muy similares. El rango de conexiones según el momento del día se ha establecido en un margen de diez minutos centrado en el instante en el que se produce la detección de un posible ataque para poder operar con más información y ser más eficaces a la hora de encontrar coincidencias entre la Snapshot y el Profile.

Una vez obtenido el perfil se pasa a la fase de comparar las coincidencias de los parámetros entre las distintas conexiones de la Snapshot y el Profile. A partir de esos resultados nos quedamos con las coincidencias



máximas de cada conexión de la Snapshot, de las cuales finalmente se obtendrá un número del cero al treinta proveniente de la media y la desviación típica de dichas coincidencias máximas, ya que lo que se pretende como un objetivo final es obtener la anomalía general de la red.

Por otro lado se calcula la anomalía de carga observando el tamaño de los paquetes y la velocidad que está soportando la red en comparación con la información indicada en el perfil. Para comparar los tamaños de los paquetes el perfil ofrece el número de paquetes y la información intercambiada en toda la conexión en los dos sentidos, por lo que para compararlo con el paquete capturado en realtime, el total de información intercambiada hay que dividirla entre el número de paquetes. En este punto es importante distinguir entre los dos sentidos de la comunicación, que vendrá establecido por el sentido que tenga algún dato en la conexión capturada por la Snapshot. Finalmente se establece el grado de anomalía de cada paquete en función de una densidad de probabilidad definida por la media y la varianza, y el grado de anomalía general de este punto será la media de las anomalías de los distintos paquetes capturados, que tendrá un valor del cero al diez.

El cálculo de la anomalía en la velocidad soportada por la red también es algo compleja, ya que también involucra el uso de una función de probabilidad para tener en cuenta tanto la media como la varianza de la carga. Para establecer la función el procedimiento que se ha seguido ha sido la división de las conexiones del perfil en grupos separados según los distintos minutos del día al que perteneciesen. Dentro de cada grupo

se calcula la carga media, basándose en la duración, tiempo y bytes totales (incluyendo cabeceras) de cada conexión. Esto tiene como resultado un conjunto de valores que representan la carga media de la red en cada uno de los minutos, que se usarán para definir la función de probabilidad mediante el uso de su media y desviación. El resultado vendrá dado por la comparación de la carga en el momento de la toma de la Snapshot mediante un método donde se tiene en cuenta la duración total de la misma, y comparándola con la función mencionada del perfil da como resultado un número del cero al diez representando la anomalía en la carga.

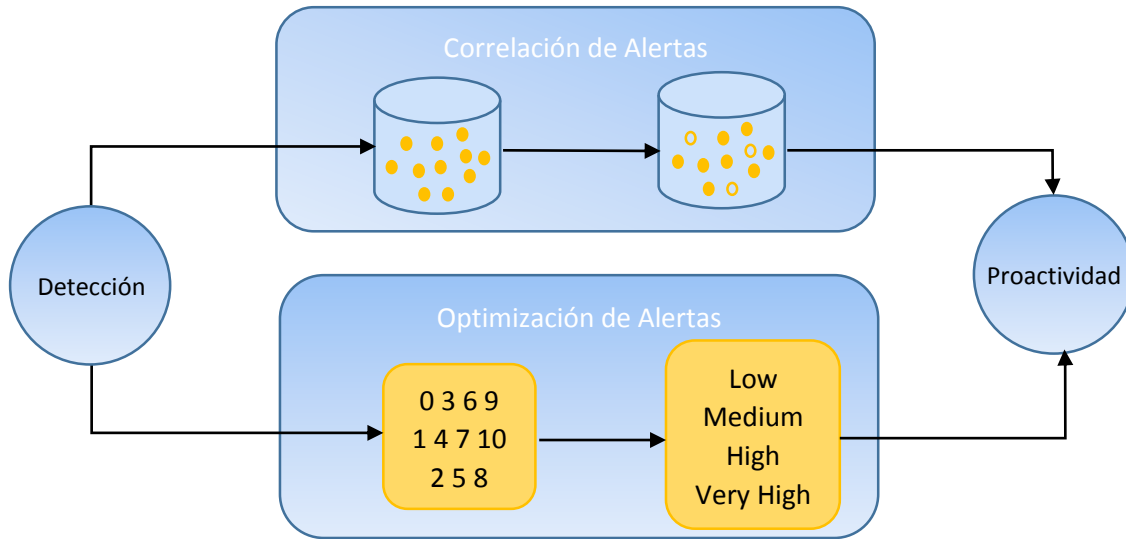
La clase encargada de las estadísticas, que realmente esa probabilidad será la confianza o desconfianza en la Snapshot, es la clase LoadStatistics, con la cual no sólo definimos la función sino que somos capaz de realizar una estimación de su integral mediante el uso de una tablas variables, que para el propósito de este proyecto proporcionan un margen de error más que aceptable.

Para obtener el número final que representa la anomalía del contexto de red, simplemente es necesario combinar de forma lineal estas tres anomalías parciales de tal forma que el resultado total sea un número del 0 al 10, siendo 10 la máxima anomalía posible.

Una vez obtenidos los valores de la anomalía que van de entre 0 a 10, el optimizador de alertas mantiene todas las alertas pero aumenta un poco la prioridad de la alerta obtenida del proceso de correlación. A

continuación se representa gráficamente como está estructurado internamente el módulo de optimización de alertas.

Figura 24. Representación gráfica del componente de Optimización de Alertas



Fuente. Elaboración Propia

Para lograr la optimización de las alertas, se utiliza una función para aumentar la severidad de las mismas, ya que si se utilizan son modificar podrían generar un falso negativo en la proactividad. Por lo tanto se necesita una función para aumentar la gravedad de la alerta de manera exponencial. Esta función se inicia sin modificar la anomalía previamente detectada para cada alerta. La fórmula a utilizar es la siguiente tomada de (SHAMELI-SENDI, EZZATI-JIVAN, JABBARIFAR, & DAGENAIS, 2012).

$$Alert.severity = Alert.severity * e^{\frac{F*N}{K*A}}$$

$$1.25 \leq K$$

$$1 \leq F \leq 100$$

$$1 \leq A, N$$
(1)

Dónde: N es la frecuencia de la alerta, F es el efecto de la alerta, A es el numero aceptable de alertas por día, K es un parámetro constante.



Proactividad

Es aquí donde los algoritmos de Modelos Ocultos de Markov (HMM) son utilizados. Uno de las limitaciones que surgió al construir este módulo fue la escasa información en español acerca de Modelos Ocultos de Markov y el no haber encontrado persona alguna que pueda explicar en qué consiste un HMM, sin embargo se buscó apoyo fuera del país logrando establecer contacto con el Dr. Anastacio Antolino Hernández quien en su Tesis Doctoral “Detección de Anomalías en Redes de Computo Utilizando Modelos Ocultos de Markov y Programación Evolutiva” uso los HMM y los optimizo usando algoritmos genéticos. Gracias su apoyo se logró comprender al detalle que era un HMM, tipos, funcionamiento, además actualmente brinda su asesoría a este trabajo de investigación.

Los HMM funcionan bien cuando se trata de entradas streaming. Un HMM es un modelo estadístico de Markov con estado no observada. Como otro punto de vista, HMM es un modelo más simple de red bayesiana dinámica. En HMM, los estados no son visibles pero la salida depende de los estados que son visibles. Es rápido y puede ser útil para evaluar el riesgo y predecir futuros ataques en los sistemas de detección de intrusos.

En los párrafos siguientes, se describen los elementos de HMM. Un HMM se caracteriza por lo siguiente:

- 1. Estados:** El sistema se supone que está en uno de los siguientes estados. Los estados utilizados en este trabajo son similares a los estados tomados por (K. Haslum, A. Abraham, & S. Knapskog, DIPS: A framework for distributed intrusion prediction and



prevention using hidden markov models and online fuzzy risk assessment, 2007)

Normal: indica que el sistema está funcionando bien y no hay ninguna actividad maliciosa o cualquier intento de entrar en el sistema.

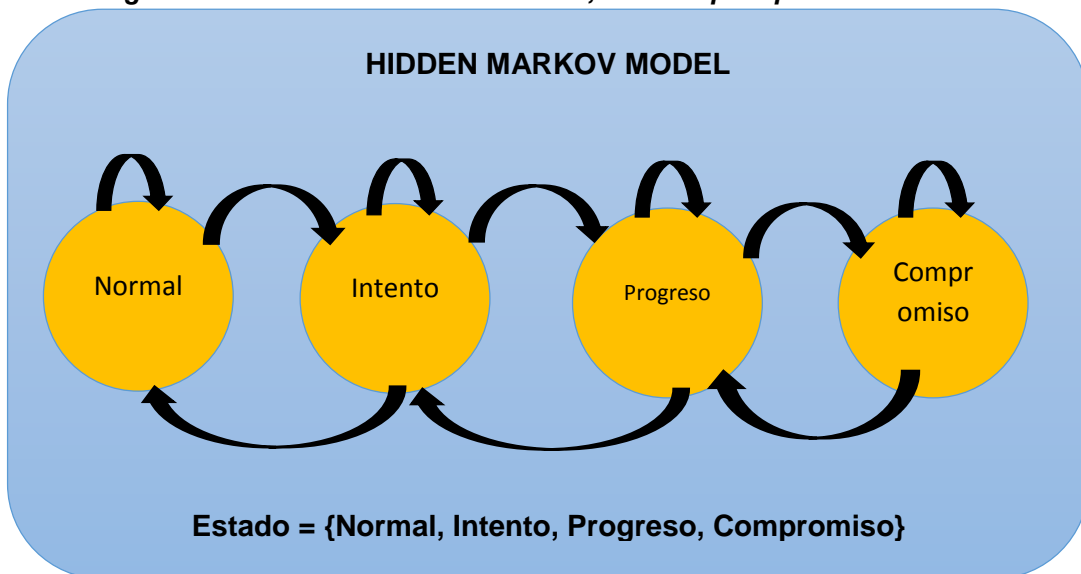
Intento: indica que las actividades maliciosas se intentan contra el sistema

Progreso: lo que indica que la intrusión se ha iniciado y ahora está progresando

Compromiso: lo que indica que la intrusión ha comprometido con éxito el sistema.

Se usa N, I, P, C para representarlos así $S_i = \{S_1=N, S_2=A, S_3=P, S_4=C\}$.

Figura 25. Modelos Ocultos de Markov, estados para proactividad



Fuente. Elaboración Propia



2. Observaciones: $O_i = \{O_1, O_2, O_3, \dots, O_n\}$ las observaciones son la producción real del sistema que se está modelando. Las observaciones hacen que el modelo del sistema se mueva entre los estados. En este caso, las alertas de componentes de detección son nuestras observaciones. Consideramos la gravedad de alertas como la observación. Cada alerta tiene tres prioridades: bajo, medio y alto. Sin embargo, no utilizamos la verdadera gravedad de observaciones. Después de recibir la gravedad real que tiene tres niveles, mapeamos que después de la optimización de alerta a las cuatro prioridades: bajo, medio, alto, muy alto. En la figura 13, se puede ver nuestro modelo para asignar la verdadera gravedad de la mayor gravedad mediante una función exponencial.

3. Probabilidad de Transición de Estado (P): La matriz de probabilidad de transición de estados describe la probabilidad de moverse entre los estados.

4. Probabilidad de transición de observación (Q): La matriz de probabilidad de transición de observación describe la probabilidad de moverse entre las observaciones.

5. Distribución del Estado Inicial: Describe la probabilidad de estados cuando inicia nuestro el sistema

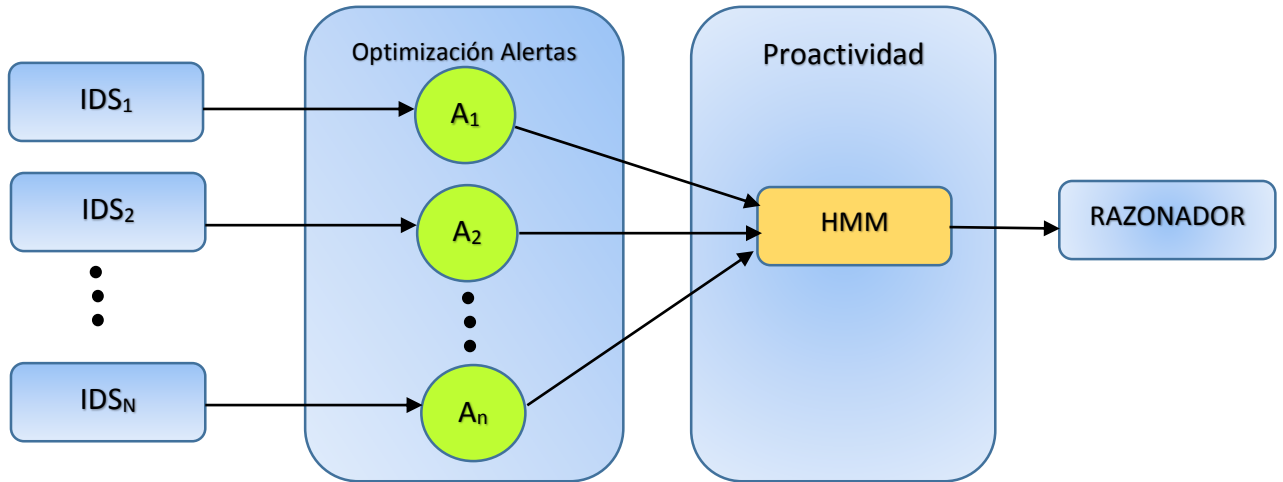
Desde el punto de vista de implementación el módulo de proactividad ha sido construido primero bajo la plataforma Mathematica Versión 8 a fin de hacer pruebas de manera rápida y comprobar si era posible evolucionar los HMM usando programación evolutiva, lo cual fue exitosa.



Ahora vamos a describir el módulo de proactividad en detalle. Como se ya ha mencionado todos los componentes de detección envían alertas al razonador y este solicita al módulo de contexto para determinar el grado de la alerta. Una vez obtenidos los valores de anomalía del contexto de red y de sistemas, el componente de optimización de alerta aumenta la gravedad de alerta utilizando una función exponencial. El aumento de la gravedad de alertas se envía al módulo de proactividad como la observación. Para cada observación, el HMM creado automáticamente a través de la programación evolutiva mueve entre los estados y la probabilidad de estar en cada estado se actualizará. El cálculo necesario para actualizar el estado de distribución se basa en la Ecuación 19 y 27 de (Rabiner, L. R., A tutorial on hidden markov models and selected applications in speech recognition, 1989) y el algoritmo 1 de (K. Haslum, M.E.G. Moe, & S.J. Knapskog, Real time intrusion prevention and security analysis of networks using HMMs, 2008).

En la siguiente figura se muestra la estructura del módulo de proactividad

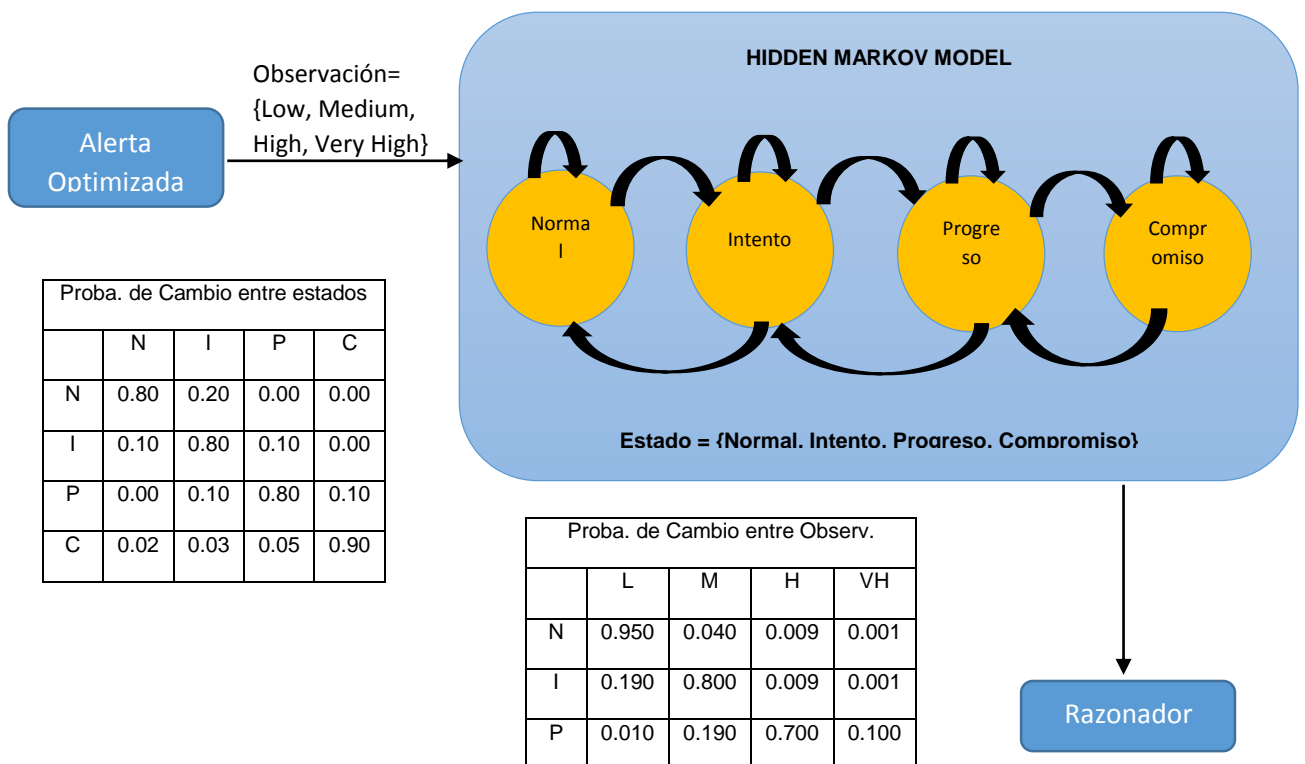
Figura 26. Representación gráfica del módulo de proactividad



Fuente. Elaboración Propia

La siguiente figura muestra la forma en que el HMM genera automáticamente su tabla de transiciones.

Figura 28. Algoritmo Proactividad - HMM



Fuente. Elaboración Propia

C	0.010	0.040	0.200	0.750
---	-------	-------	-------	-------

Receptor de Contexto

Componente de la arquitectura que recibe la información relativa a la anomalía en el contexto de red y de sistemas procedentes de los módulos encargados de obtenerla, y mapea dicha información a los conceptos equivalentes definidos en la ontología. Una vez que la información ha sido convertida a lenguaje OWL2, este componente devuelve el control al Razonador ya la vez es enviado al módulo de predicción para que comience el proceso de inferencia propiamente dicho.

Desde el punto de vista de su implementación, este componente es un programa Java que utiliza el API de Jena para su interacción con la ontología de respuesta a intrusiones. A modo de ejemplo, las siguientes líneas de código muestran cómo el Receptor de Contexto crea un nuevo ejemplar de la clase ProbabilidadIntrusion de la ontología y asigna el valor correspondiente a sus atributos, para lo que utiliza la información de proporcionada por el contexto de red relativa al grado de anomalía presente en la red en momento de intrusión.

```

OntClass    class_NetworkContext    =    ontologyModel.getOntClass(PREFIXIND+
"NetworkContext");
    String idnc = "network"+ide;
    Individual resNuevo = ontologyModel.createIndividual(PREFIXIND + idnc,
class_NetworkContext);
    Property prop;
    ejemplarContextoRedNuevo = idnc;
    if(Integer.toString(anomaly) != null){
        Literal netanomaly = ontologyModel.createTypedLiteral(anomaly);
        prop = ontologyModel.getProperty(PREFIXIND + "networkAnomaly");
        resNuevo.setPropertyValue(prop, netanomaly);
    }
    
```



Ontología de Respuesta a Intrusiones

Componente fundamental en la arquitectura del AIRS Proactivo, cuya función es definir y especificar formalmente toda la información necesaria en el proceso de inferencia de la respuesta óptima frente a una intrusión llevado a cabo por el AIRS. Así pues, la ontología define formalmente conceptos como alerta de intrusión, tipo de amenaza, acción de respuesta, contexto de red y sistemas, etc.

El lenguaje utilizado para especificar la ontología es el lenguaje estándar de la Web Semántica OWL2. La definición y especificación de la ontología es primordial para el correcto funcionamiento del AIRS ya que en ella reside toda la información utilizada por el Razonador para inferir la respuesta óptima.

Se puede considerar, que junto con las Políticas o reglas y el propio Razonador, es el principal componente del sistema.

Respecto a su integración en la arquitectura del AIRS Proactivo desde el punto de vista de su interacción con el Razonador, cabe especificar que en el prototipo implementado el Razonador utiliza el API de Jena para leer y escribir en la ontología de respuesta a intrusiones. Cuando el Razonador se arranca, la primera acción que realiza es leer la ontología y guardar una copia de la misma como un objeto de la clase `OntModel`, para lo que ejecuta lo siguiente:

```
System.out.println("Leyendo la ontología ...");
synchronized (ontologyindividuos) {
    OntModel ontologyModel =
    ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
```



```
RDFReader inf_modelReader = ontologyModel.getReader();
inf_modelReader.setProperty("WARN_UNQUALIFIED_RDF_ATTRIBUTE",
"EM_IGNORE");
inf_modelReader.read(ontologyModel, ontologyindividuos);
}
```

Una vez finalizado el proceso de inferencia de la respuesta óptima y calculado el éxito de la acción de respuesta, el Razonador actualiza la información obtenida en la ontología de respuesta a intrusiones, para lo que utiliza la API de Jena.

Métricas

Las métricas (políticas o reglas) tienen el objetivo de especificar el comportamiento del AIRS, es decir, de gobernar el proceso de inferencia de la respuesta óptima llevado a cabo por el Razonador.

Se ha elegido el lenguaje SWRL (I. Horrocks, y otros, 2004) como lenguaje para la especificación de las políticas y reglas de la arquitectura propuesta.

Las reglas constituyen las pautas en función de las cuales y partiendo de un conjunto de conocimientos previos, el Razonador infiere la respuesta más conveniente para una determinada intrusión, en un determinado lugar, y en un determinado momento. La contribución de este componente al funcionamiento del sistema es, por tanto, crucial.

Añadir además que estas reglas SWRL permite modelar y especificar aquellas métricas de seguridad propuestas que impongan pautas al comportamiento del AIRS Proactivo. En un AIRS hay muchos parámetros susceptibles de ser medidos, como por ejemplo, la confianza en el IDS, el



impacto de la intrusión, el coste, el impacto y la eficacia de la respuesta, la elección de la respuesta óptima, la fiabilidad de la alerta de intrusión, etc. Para obtener el valor de estos parámetros es necesario utilizar métricas. Uno de los pasos importantes en la definición e implementación de la arquitectura del sistema de respuesta propuesta, es la definición de las métricas de seguridad necesarias para obtener aquellos parámetros involucrados en el proceso de inferencia de la respuesta óptima que requieren ser medidos. Se distingue entre dos tipos de métricas: Aquellas que se ejecutan de manera independiente al proceso de inferencia de la respuesta óptima. Este tipo de métricas pueden ser ejecutadas por el propio sistema de forma automática o por el administrador de forma manual, y no requieren de su especificación mediante reglas SWRL porque no forman parte del conjunto de políticas de la arquitectura del sistema. Ejemplo de estas métricas es la métrica utilizada para obtener el impacto de la intrusión en el sistema y aquellas que se ejecutan como parte del proceso de inferencia de la respuesta óptima, y que determinan el comportamiento del AIRS. Este tipo de métrica siempre es ejecutada de forma automática por el Razonador del sistema en tiempo de inferencia, y es necesario especificarlas mediante el lenguaje de reglas SWRL. Estas métricas dan lugar a un conjunto de reglas o políticas que utiliza el Razonador para inferir la respuesta óptima. Ejemplo de estas métricas es la métrica utilizada para la elección de la respuesta óptima.

Razonador

Es el componente principal de la arquitectura, cuya función es inferir la respuesta óptima ante una intrusión detectada por alguno de los IDSs existentes en la red de la organización. Teniendo en cuenta las políticas definidas previamente por el administrador del sistema y los ejemplares de la ontología que representan la información acerca de las respuestas, la información de contexto, alertas de intrusiones, etc., el razonador ejecuta el proceso de inferencia para determinar la respuesta óptima a una intrusión específica en un lugar y tiempo específicos. Para ello, es necesario un mecanismo que permita inferir información a partir del conocimiento almacenado en la base de conocimiento, como son los razonadores semánticos. Hoy en día, existen diferentes razonadores semánticos que pueden ser utilizados como el núcleo del AIRS. Tras un análisis comparativo realizado, se elige Pellet como razonador semántico a utilizar en la arquitectura de AIRS propuesta.

En este punto es necesario aclarar que el componente Razonador presente en la arquitectura del sistema, no es solamente el razonador semántico Pellet. Es decir, el término Razonador incluye al propio Razonador semántico Pellet, además de código escrito Java que permite realizar una serie de funciones entre las que destacan:

Comprobar la similitud o continuación de una alerta de intrusión: cuando recibe una alerta de intrusión en formato común procedente del Receptor de Alertas, comprueba si la nueva alerta recibida se corresponde con una alerta ya recibida por el sistema, o si se corresponde con una alerta de un



ataque ya existente. Para ello, se utiliza el lenguaje SPARQL que permite realizar consultas sobre la información almacenada en la ontología, como se detallara en la implementación del prototipo del sistema.

- ✓ Invocar a los módulos de contexto de red y de sistemas para obtener el grado de anomalía en un determinado momento.
- ✓ Inferir las respuestas recomendadas y la respuesta óptima: esta función es su función principal, y es en este punto donde se hace uso del razonador semántico Pellet para razonar sobre el domino modelado en la ontología utilizando las reglas y políticas definidas, y de esta forma inferir nueva información.
- ✓ Extraer el nuevo conocimiento inferido. Para ello, se hace uso de Jena, un API de Java soportado por Pellet, que permite extraer datos de grafos RDF.
- ✓ Invocar al Ejecutor de Acciones de Respuesta una vez que la respuesta óptima ha sido inferida, proporcionándole los parámetros requeridos para desplegarla.
- ✓ Invocar al Evaluador del éxito de Acciones de Respuesta para evaluar el éxito de la acción una vez finalizada su ejecución, proporcionándole los parámetros requeridos por dicho componente.
- ✓ Obtener información de la ontología de respuesta a intrusiones mediante Jena.
- ✓ Actualizar la información inferida (respuesta óptima, éxito de la respuesta...) en el fichero owl correspondiente a la ontología de respuesta a intrusiones, de forma que esta nueva información se tenga en cuenta en posteriores inferencias. Para escribir en el fichero owl se utiliza Jena.

- ✓ Actualizar el nuevo resultado obtenido en el histórico de intrusiones y respuestas correspondiente, de cara a un posible funcionamiento en entornos colaborativos.

Como se observa en la arquitectura, este componente es el centro de la misma, es el núcleo del AIRS Proactivo. El resto de componentes de la arquitectura interactúan con él, bien proporcionándole entradas, bien haciendo uso de los resultados inferidos por el Razonador.

Desde el punto de vista de su implementación es un programa Java multihebra, es decir, permite ejecutar varias inferencias en paralelo, lo que permite aumentar su rendimiento. El valor del número de inferencias puede ser configurado por el administrador, y puede ser tan alto como se desee. No obstante, no se recomienda utilizar un número muy elevado de hebras puesto que podría dar lugar a problemas de inconsistencias con la ontología. En el prototipo desarrollado se utilizan 4 hebras.

Cada vez que el Receptor de Alertas recibe una intrusión procedente de un IDS, se crea una instancia del Razonador (clase Java OntAIRS), tras lo que se inicializa dicha instancia mediante la ejecución de las siguientes líneas de código:

```
OntModel inferedModel =
ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC,
null);
inferedModel.read(ontologyrules);
inferedModel.add(ontologyModel);
inferedModel.prepare();
```

En primer lugar se crea una instancia de la clase OntModel indicando que el razonador semántico utilizado es Pellet. A continuación, se lee el fichero que

contiene las reglas SWRL especificadas y se asocia al modelo creado, tras lo que se añade otro modelo que contiene la información contenida en el momento específico en la ontología de respuesta a intrusiones (variable `ontologyModel`). Como último paso de la fase de inicialización del Razonador se ejecuta el método `prepare ()`, que permite que el razonador semántico infiera nuevo conocimiento a partir de la base de hechos y las reglas SWRL.

Una vez inicializado, el Razonador comprueba si la alerta de intrusión recibida se refiere a una intrusión ya existente o es una alerta nueva, tras lo que invoca a los módulos de contexto. A continuación, infiere el conjunto de respuestas recomendadas y óptimas para lo que utiliza la información contenida en la ontología de respuesta a intrusiones y las reglas SWRL. Una vez inferida la respuesta óptima, invoca al módulo de ejecución de respuestas y tras finalizar su ejecución, invoca al sistema de evaluación del éxito de una acción de respuesta. Por último, actualiza el valor de RTE asociado a la acción de respuesta desplegada en la ontología para que sea tenido en cuenta en posteriores inferencias.



Catálogo de Acciones de Respuesta

Su función es proporcionar a los componentes de la arquitectura que lo requieran el conjunto o listado de acciones de respuesta específicas, que están disponibles para reaccionar frente a una intrusión con el fin de mitigarla o reducir el daño causado. Estas respuestas se ejecutan a través de diferentes componentes de seguridad (routers, cortafuegos, IDSs, etc.) o a través de aplicaciones del sistema operativo subyacente (tcpwrapper, cortafuego local, comandos locales, etc.).

Ejecutor de Acciones de Respuesta

Componente de la arquitectura encargado de poner en marcha mecanismos reales para ejecutar la respuesta inferida por el Razonador, proporcionando los argumentos e instrucciones necesarias a los dispositivos de seguridad del sistema encargados de hacer efectiva la respuesta. Para ello, consulta el catálogo de respuestas disponible, Una vez inferida la respuesta óptima, el Razonador, invoca a este componente, proporcionándole los parámetros necesarios para la ejecución de la respuesta específica.

Evaluador del éxito de Acciones de Respuesta

Su objetivo es evaluar de forma automática si la respuesta ejecutada ha conseguido el objetivo de mitigar la intrusión o reducir al mínimo su impacto, o por el contrario, no ha tenido efecto sobre el ataque.

El Razonador es el responsable de invocar a este componente una vez que ha recibido un código de que la respuesta ha finalizado su ejecución, código enviado por el Ejecutor de Acciones de Respuesta, no es tarea sencilla medir



de forma automática el éxito o eficacia de una respuesta. La metodología utilizada en la arquitectura de AIRS propuesto, consiste en comparar el grado de anomalía existente en el contexto de la red y del sistema comprometido, antes y después de la ejecución de la respuesta, para lo que se usará técnicas de la teoría de la información, como es el concepto de entropía. La definición y especificación de una metodología de evaluación del éxito de una acción de respuesta, así como el diseño e implementación de una arquitectura que aplique dicha metodología.

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

1. La programación evolutiva resulto ser más eficiente que los Algoritmos Forward-Backward, Viterbi y Baum-Welch comúnmente usados en las investigaciones estudiadas para modelar los HMM. Con el algoritmo de Baum-Welch se concurría en el problema de poder converger en un máximo local, no pudiendo de esta manera llegar a entrenar adecuadamente al modelo HMM.
2. Los HMMs generados con programación evolutiva tuvieron mucha mayor eficiencia que los entrenados con el algoritmo de Baum-Welch. El algoritmo de Baum-Welch solo refina u optimiza los valores iniciales del HMM, en cambio los HMMs generados con programación evolutiva no requieren de conocimiento previo proporcionan mejores resultados.
3. **Un punto importante en este trabajo fue el proporcionar solución alternativa al problema de ajuste y optimización de parámetros de un HMM dada la serie de observación, con este trabajo se diseña y entrena un HMM con solo la secuencia de observación, encontrando los mejores parámetros para el HMM.**
4. El sistema evolutivo es capaz de modelar HMMs con secuencias de valores de variables aleatorias continuas, sin necesidad de discretizar los valores originales. Otras herramientas, como Jahmm (Java Tool para HMMs), solo trabaja con valores discretos.



5. En una primera comparación entre los HMMs Univariados y un HMM-BW (BaumWelch), optimizado con el Algoritmo de BW, nos proporcionó un mejor análisis de detección de las anomalías el modelo de HMMs generados con Programación Evolutiva (EP) que el optimizado con BW.
6. La propuesta de utilizar Modelos Ocultos de Markov y programación evolutiva para evolucionarlos, y utilizar los modelos evolucionados en la optimización de las alertas, fue como se esperaba, ya que las pruebas que se han hecho han resultado ser capaces de hacer inferir la respuesta correcta al AIRS.
7. Se ha evidenciado que los HMM de la mano de la Programación Evolutiva pueden ser utilizados para la detección exitosa de anomalías en redes

6.2 Recomendaciones

1. Un análisis de los riesgos que coexisten en redes TCP/IP serviría para poder establecer estrategias que logren contra restar el paso de los ataques en redes de computo.
2. Si se utiliza el algoritmo de Baum-Welch como técnica de optimización de un HMM, este debe usarse solo en la población inicial generada aleatoriamente, y a partir de allí comenzar la evolución de los HMMs con los algoritmos propuestos.
3. Optimizar el tiempo de procesamiento basado en secuencias cortas de observación o la utilización de otras funciones evaluadoras de aptitud..
4. Se sugiere como trabajo futuro migrar el código de mathematica a otro lenguaje de programación como C o java a fin de poder integrarlo al AIRS estudiado.
5. Como trabajo futuro se debe Integrar la totalidad de los módulos del AIRS estudiado a fin de lograr una aplicación robusta y funcional que pueda actuar proactivamente ante cualquier ataque que se pudiera presentar en una red, se sugiere continuar la línea de investigación trabajando bajo Linux.

Referencias

Anner, S. (2011). *Intrusion Detection System*.

Antolino Hernandez, A. (2011, FEBRERO). *Detección de Anomalías en Redes de Computo Utilizando Modelos Ocultos de Markov y Programacion Evolutiva*. SAN NICOLAS DE HIDALGO, MEXICO.

ANTOLINO HERNANDEZ, A. (2011, Febrero). DETECCION DE ANOMALIAS EN REDES DE COMPUTO UTILIZANDO MODELOS OCULTOS DE MARKOV Y PROGRAMACION EVOLUTIVA. Morelia, Michoacan, México.

B. Zhu, & A. A. Ghorbani. (2008). Alert Correlation for extracting attack satrategies. *International Journal of Network Security Vol. 3*, (pp. 244-258). Montreal, Canada.

Barrera, J. , & Flores, J. (2008).

BERGASA P., & LUIS M. (n.d.). *Introducción a los Modelos Ocultos de Markov*. Retrieved from Departamento de Electrónica, Material de la Cátedra de Métodos Estadísticos en Ciencias de la Vida, Universidad de Alcalá:
<http://www.bioingenieria.edu.ar/academica/catedras/metestad/Introduccion%20al%20Modelo%20oculto%20de%20Markov.pdf>

Bishop, M. (2003). *Computer Security: Art and Science*. U.S.A: UK: Addison-Wesley Publishing Co.

Blunsom Phil. (2004). *Hidden Markov Model*. Retrieved from
<http://www.cs.mu.oz.au/460/2004/materials/hmm-tutorial.pdf>



- C. Kruegel, F. Valeur, & G. Vigna. (2005). Aert Correation, in Intrusion, Detection and Correlation. New York: Springer.
- CAPPÉ OLIVIER, M. E. (2005). Inference in Hidden Markov Models.
- Cappé, O., Moulines, E., & Rydén, T. (2005). *Inference in Hidden Markov Models*,.
- Chau, C., Kwong, S., & Diu, C., et al. (1997). *Optimization of hmm by a genetic algorithm. Acoustics, Speech, and Signal Processing*.
- Chien, J.-T. , & Liao, C.-P. (2008). *Maximum confidence hidden markov modeling for face recognition*.
- CONVERY SEAN, & MILLER DARRIN. (2004). *IPv6 and IPv4 Threat Comparison and Best-Practice Evaluation*. Retrieved from CISCO White Paper, CISCO Systems:
http://www.cisco.com/security_services/ciag/documents/v6-v4-threats.pdf
- Dan, S. , & YanLing, S. (2010). Detection of network intrusion based on a hmm. *Multimedia and Information Technology (MMIT)*.
- De Floriani, L., & Spagnuolo, M. (2008). *Shape Analysis and Structuring*.
Springers.
- DEBAR, H., CURRY, D. A., & FEINSTEIN, B. S. (2007). *THE INTRUSION DETECTION MESSAGE EXCHANGE FORMAT*. IDMEF.
- Diego Evin, Alejandro Hadad, Mauro Martina, & Bartolomé Drozdowicz. (2010, Diciembre). *Predicción de estados de Hipotensión empleando Modelos Ocultos deMarkov*. Colombia.



EC-Council. (n.d.). *Certified Ethical Hacker*. Retrieved from

<http://www.eccouncil.org/ceh.htm>

Ephraim, Y. , & Merhav, N. (2002). *Hidden markov processes*. *IEEE*

TRANSACTIONS ON INFORMATION THEORY.

F. Galan, D. Fernandez, J.E. Lopez de Vergara, & R. Casellas. (2010). Using

model driven architecture for technology-independent scenario

configuration in networking testbeds. *IEEE Commun Mag 2010*, 132-141.

Florez, R., & Fernandez, J. M. (2008). *Las Redes Neuronales Artificiales*,

Fundamentos teoricos y aplicaciones prácticas. Netbiblo.

Ghosh, A., Wanken, J., & Charron, F. (1998). Detecting anomalous and

unknown intrusions against programs. *omputer Security Applications*

Conference.

GRAVES , K. (2007). CEH Official Certified Ethical Hacker. Review Guide.

Wiley Publishing.

Guaman Loachamin, D. S. (2013). Trabajo de Fin de Máster. *Propuesta de*

Arquitectura e Implementación de un Módulo de Ejecución de Acciones

de Respuesta para un Sistema Autónomo de Respuesta a Intrusiones

basado en Ontologías. Madrid, Esaña.

I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosf, & M. Dean.

(2004). *SWRL: A Semantic Web Rule Language Combining OWL and*

RuleML.



Institute for Security and Open Methodolog, I. (n.d.). OSSTMM. Retrieved from OSSTMM: <http://isecom.securenetsltd.com/OSSTMM.es.2.1.pdf>

INTERNATIONAL COUNCIL OF ELECTRONIC COMMERCE CONSULTANTS. (n.d.). Ethical Hacking. Official Course Material. 2004.

Jabbour, G., & Maldonado, J. (2009, Junio). Predicción de Índices Bursátiles mediante un Sistema Híbrido basado en Modelos Ocultos de Markov y Redes Neuronales Artificiales. Mérida, Venezuela.

Jacob, C. (2001). *Illustrating Evolutionary Computation with Mathematica*.

Jemili, F., Zaghdoud, M., & Ben Ahmed, M. (2007). A framework for an adaptive intrusion detection system using bayesian network. *Intelligence and*.

K. Haslum, A. Abraham, & S. Knapskog. (2007). DIPS: A framework for distributed intrusion prediction and prevention using hidden markov models and online fuzzy risk assessment. *In Third International Symposium on Information Assurance and Security*, (pp. 183-188).

K. Haslum, M.E.G. Moe, & S.J. Knapskog. (2008). Real time intrusion prevention and security analysis of networks ussing HMMs. *33rd IEEE Conference on Local Computer Networks*. Montreal.

K. Scarfone, & P. Mell. (2007). Guide to Intrusion Detection and Prevention Systems. Tecnica Report NIST SP800-94. National Institute od Standars and Tecnology.

Kaspersky, L. (2014). *Kaspersky-Lab-los-ataques-dirigidos-estan-aumentando-ano-tras-ano.note.aspx*. Retrieved from



<http://www.tyngovernments.com/379642-Kaspersky-Lab-los-ataques-dirigidos-estan-aumentando-ano-tras-ano.note.aspx>

KLEVINSKY , T., LALIBERTE , S., & GUPTA , A. (2002). *Hack I.T. Security through Penetration Testing*. Addison Wesley.

Korayem, M., Badr, A., & Farag, I. (2007). *Optimizing hidden markov models using genetic algorithms and artificial immune systems*.

LAWRENCE, M., & GREGORY PETER. (2007). *CISSP for Dummies, 2nd Edition*. Jhon Wiley & Sons.

Li, Y., Wang, R., Xu, J., Yang, G., & Zhao, B. (2009). Intrusion detection method based on fuzzy hidden markov model. Washington, USA: IEEE Computer Society.

López, M.-E. et al. (2005). *A navigation system for assistant robots using visually augmented pomdps*. *Autonomous Robots*,.

MARCHETTE , D. (2001). *Computer Intrusion Detection and Network Monitoring. A Statistical Viewpoint*. Springer-Verlag.

Martínez, M. (2009). *La influencia de Darwin en el pensamiento científico contemporáneo*. Retrieved from <http://www.uv.mx/cienciahombre/revistae/vol22num3/articulos/darwin/index.html>

Mateos Lanchas, V. (2013, Noviembre). Tesis Doctoral. *Contribucion a la Automatización de Sistema de Respuesta Frente a Intrusiones Mediante Ontologías*. Madrid, España.



MATEOS, V., VILLAGRÁ, & ROMERO, F. (2010). ONTOLOGIES-BASED AUTOMATED INTRUSION RESPONSE SYSTEM. *COMPUTACIONAL INTELLIGENCE IN SECURITY FOR INFORMATION SYSTEMS 2010*, 99-106.

Mesa, A. (2010, Noviembre). Tesis de Maestría en Ingeniería Matemática . *Modelos markovianos para secuencias y aplicaciones a la predicción de genes*. Montevideo, Uruguay.

Miners, B. , & Basir, O. (2005). *Dynamic facial expression recognition using fuzzy hidden markov models*.

MIT Lincoln Laboratory. (n.d.). 2000 DARPA Intrusion Detection Scenario Specific Data Sets, 2000.

N. B. Anuar, M. Papadaki, S. Furnell, & N. Clarke. (2010). An investigation and survey of response options for Intrusion Response Systems (IRSs).

Nicholl, P., Amira, A., Boucha, D., & Perrot, R. H. (2008). *A statistical multiresolution approach for face recognition using structural hidden markov models*. EURASIP Journal on Advances in Signal Processing, ACM,.

OSSEC. (2013, 03 01). *OSSEC V2.7.0*. Retrieved from DOCUMENTACION LOG SAMPLES.: http://www.ossec.net/doc/log_samples/index.html

Pachter, L. et al. (2001). *Applications of generalized pair hidden markov models to alignment and gene finding problem*. RECOMB .

- Phil Blunsom. (2004, Agosto 19). Hidden Markov Models. *Hidden Markov Models*.
- Ponce Cruz, P. (2010). *Inteligencia Artificial con aplicaciones a la ingeniería*. Alfaomega.
- Rabiner, L. R. , & Juang, B. H. (1986). *An introduction to hidden markov models*. IEEE ASSP Magazine.
- Rabiner, L. R. (1989). *A tutorial on hidden markov models and selected applications in speech recognition*. Proceedings of the IEEE.
- RECLAMO. (2013). *RECLAMO*. Retrieved from Virtual and Collaborative Honeynets based on Trust Management and Autonomous Systems applied to Intrusion Management: <http://reclamo.inf.um.es/>
- Robayo Santana, E. L. (2009). Tesis de Maestría. *Detección de Intrusos en Redes de Telecomunicaciones IP*. Bogota, Colombia.
- Rodríguez , A., & R. Darío. (2007). Sistema Adaptativo para la Predicción de Incendios Forestales Basado en Estrategias Estadístico-Evolutivas. Barcelona, España.
- Romero Morales, C., Ventura Soto, S., & De Castro, C. (2009). Aplicacion de Algoritmos Evolutivos como Técnica de Minería de Datos para la Mejora de Cursos Hipermedia Adaptativos basados en Web. Cordova, España.
- Russell, S., & Norvig, P. (2003). *Artificial Intelligence, A Modern Approach*. Prentice Hall.



SANS , I. (n.d.). *SANS Security 560: Network Penetration Testing and Ethical Hacking*. Retrieved from

<http://www.sans.org/training/description.php?mid=937>

SHAMELI-SENDI, A., EZZATI-JIVAN, N., JABBARIFAR, M., & DAGENAIS, M. (2012). INTRUSION RESPONSE SYSTEMS: SURVEY AND TAXONOMY. *INT J COMPUT SCI NETWORK SECUR (IJCSNS)*, 1-14.

SNORT. (2013, 03 01). *SNORT USERS MANUAL 2.9.3*. Retrieved from http://www.snort.org/assets/166/snort_manual.pdf.

Stakhanova, N., Basu, S., & Wong , J. (2007). A TAXONOMY OF INTRUSION RESPONSE SYSTEMS. *INTERNATIONAL JOURNAL OF INFORMATION AND COMPUTER SECURITY*, 169-184.

Stakhanova, N., Basu, S., & Wong, J. (2007). A TAXONOMY OF INTRUSION RESPONSE SYSTEMS. *INTERNATIONAL JOURNAL OF INFORMATION AND COMPUTER SECURITY*, 169-184.

STUART, M., SCAMBRAY , J., & KURTZ , G. (2003). *HACKERS. Secretos y soluciones para la seguridad de redes*. McGraw-Hill/Interamericana de España.

Symantec. (2014). *Informe sobre Amenazas a la Seguridad en Internet, Vol. 19*. Retrieved from http://www.symantec.com/es/es/security_response/publications/threatreport.jsp

Taylor, H. M., & Karlin, S. (1998). *An Introduction to Stochastic Modeling. Academic*.



- Toure, M. (1994). An interdisciplinary approach for adding knowledge to computer.
- Túpac Valdivia, Yván Jesús; Ponce Gallegos, Julio Cesar; Torres Soto, Aurora; Quezada Aguilera, Fátima Sayuri; Silva Sprock, Antonio; Martínez Flor, Ember Ubeimar; Casali, Ana;. (2014). *Inteligencia Artificial*. Arequipa-Perú: Proyecto Latin.
- VILLAGRÁ GÓNZALES, V. (2009). *SEGURIDAD EN REDES DE TELECOMUNICACION*. FUNDACION ROGELIO SEGOVIA.
- Villagra, V., Martinez Perez, G., & Mateos Lanchas, V. (2013). *RECLAMO*. Retrieved from RECLAMO: Virtual and Collaborative Honeynets based on Trust Management an Autonomous Systems applied to Intrusion Management: reclamo.inf.um.es
- Warrender, C., Forrest, S., & Pearlmutter, B. (1999). *Detecting intrusions using system calls: Alternative data models*. En *Security and Privacy*. IEEE Symposium on.
- Won, K.-J., Hamelryck, T., Prugel-Bennett, A., & Krogh, A. (2005). *Evolving hidden markov models for protein secondary structure prediction*.
- Xiuqing, C., Yongping, Z., & Jiutao, T. (2010). Hmm-based integration of multiple models for intrusion detection. *Advanced Computer Theory and Engineering*.
- Zhang, D. , & Leckie, C. (2006). An evaluation technique for network intrusion.



ZIMMERMANN, H. (n.d.). *OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection*. Retrieved from http://www.comsoc.org/livepubs/50_journals/pdf/RightsManagement_eid=136833.pdf

ANEXOS

1. Código Fuente HMM evolucionados con Programación Evolutiva (Plataforma Mathematica 8.0)

```
<<MultivariateStatistics`
ClearAll[nEdos]
```

```
(*Dada una secuencia de valores, como entrada,
devuelve de las diferencias, la menor de todas*)
Clear[minDistance];
minDistance[serie_]:=
Block[{listsort,distances,n},
  listsort = Sort[serie];
  distances=Table[Extract[listsort,i+1]-Extract[listsort,i],
    {i,Length[listsort]-1}];
  distances=Sort[distances];
  n=1;
  While[distances[[n]]== 0,n++];
  distances[[n]]
]
```

```
(*Normaliza los datos o series dadas, entre valores 0 y 1
como rango*)
Clear[normalizaData];
normalizaData[Serie_]:=
Block[{serieT=Serie,datoMin,datoMax},
  datoMin=Min[serieT];
  datoMax=Max[serieT]-datoMin;
  Table[{serieT[[i]] - datoMin}/datoMax,{i,Length[serieT]}]
]
```

```
(*Normaliza a 1, la suma de una lista de valores*)
Clear[normalTrans];
normalTrans[lista_]:= Block[{lis=lista},
  Table[lis[[i]]/Total[lis],{i,Length[lis]}]
]
```

```
(*Genera una Lista de valores reales (transiciones) de salida
entre estados,con una suma = 1. Recibe el num de Estados*)
Clear[Transitions];
Transitions[nEdos_]:=
Block[{list,i},list=Table[RandomReal[],{nEdos}];
  Table[list[[i]]/Total[list],{i,nEdos}]
]
```

```
(*Dado el valor de incremento, se calcula el nuevo valor con
la funcion de Distribucion Normal, regresa el valor calculado,
el cual incrementara o disminuira los valores de la transicion*)
Clear[calcIncrem];
calcIncrem[stepSize_]:=Block[{inc},
  inc=RandomReal[NormalDistribution[0,stepSize]];
  If[inc > 1, inc = 0.25,
  If[inc < 0, inc = -0.25]
  ];
  inc
]
```



```
(•Funcion que recibe la lista de valores de las variables y el
valor de Epsilon. La funcion regresa una lista con los limites para
cada variable, entre -epsi y +epsi*)
Clear[listLimites];
listLimites[listVars_,epsi_]:=
Block[{len = Length[listVars],lval=listVars,lNomVals},
  lNomVals = Table["x" <> ToString[i],{i,len}];
  lNomVals = ToExpression[lNomVals];
  Table[ToExpression["{" <> ToString[lNomVals[[i]]] <> "," <>
    ToString[lval[[i]]-epsi] <> "," <> ToString[lval[[i]]+epsi
    <> "}"], {i,len}]
]
```

```
(•Paso de inicializacion del algoritmo Forward. Recive el HMM,
la lista sencilla de datos (vector), y un epsilon. Retorna la
lista con las probabilidades del vector*)
Clear[Inicializacion];
Inicializacion[HMM_,vDatos_,epsi_]:=
Block[{i,nEdos},
  nEdos = getSizeHmm[HMM];
  Table[getPiEdo[HMM,i]*FuncProbMult[getMeansEdo[HMM,i],
    getCovarianzaEdo[HMM,i],vDatos,epsi],{i,nEdos}]
]
```

```
(•Paso de Induccion del algoritmo Forward. Recibe el HMM, un
dato de la serie de Tiempo, el vector Alpha, y el epsilon.
Retorna la lista con la probabilidad. Etapa Inductiva. HMM,
obervacion, vector de Alfas, epsi*)
Clear[Induccion];
Induccion[HMM_,ElemObs_,alfas_,epsi_]:=
Block[{j,i,probElemObs,alfaAux,matrizAij,nEdos,sumAlfa},
  matrizAij = getAllTrans[HMM];(*Matriz de Transiciones*)
  nEdos = getSizeHmm[HMM];
  alfaAux = Table[0,{nEdos}];
  For[j = 0, j < nEdos,
    probElemObs = FuncProbMult[getMeansEdo[HMM,j],
      getCovarianzaEdo[HMM,j],ElemObs,epsi];
    sumAlfa = Inner[#1*#2&,alfas,matrizAij[[All,j]]];
    alfaAux[[j]] = sumAlfa * probElemObs,
    j++];
  alfaAux
]
```



```
(*Funcion de evaluacion*)
Clear[Forwards];
Forwards[HMM_,seriesTime_,epsi_]:=
Block[{sTime=seriesTime,totAlp,alphas,len,t},
  len = Length[sTime]-1;
  alphas = Table[0,{len+1},{getSizeHmm[HMM]};
  (*Primer dato de la serie*)
  alphas[[1]] = Inicializacion[HMM,sTime[[1]],epsi];
  (*alphas=Induccion[HMM,sTime,alphas,epsi];*)
  Table[alphas[[t+1]]=
    Induccion[HMM,sTime[[t+1]],alphas[[t]],epsi],
    {t,len}];
  (*Validando que la probabilidad != 1*)
  totAlp = Total[alphas[[len + 1]]]; (*Probab acumulada*)
  If[totAlp >= 0.99, totAlp = 0.980009;
  totAlp
]
]
```

```
(*Funcion de Calculo de Probabilidad para un HMM. Toma el conjunto
de Series de Tiempo de las Options de iniChrome, asi como el valor
de Epsilon. Retorna el valor de probabilidad para el HMM*)
Clear[ForwardHMM];
ForwardHMM[HMM_,opts__]:=Block[{seriesT},
  seriesT = tSeries/.{opts}/.Options[initChrome];
  Forwards[HMM,seriesT,epsilon/.{opts}/.Options[initChrome]]
]
]
```

```
(*NOTA: Esta funcion solamente es una intermediaria entre la llamada
a la funcion de Evaluation y ForwardHMM, se podria omitir.
NOTA: quitar, por el momento no lo ocupo*)
Clear[HMMMut];
HMMMut[ForwardHMM][HMM_]:=Block[{auxHmm=HMM,fitness},
  fitness=ForwardHMM[auxHmm]
]
]
```

```
Options[Evaluation] = {EvaluationFunction :> Depth};
```

```
(*****)
Clear[Evaluation];
Evaluation[HMM_,opts__]:=Block[{auxHmm=HMM,fitness,funcion},
  funcion = EvaluationFunction /. {opts} /. Options[Evaluation];
  (*fitness=HMMMut[ForwardHMM][auxHmm];
  auxHmm[[1,1]]=fitness;*) (*HMM Calificado*)
  auxHmm[[1,1]]=funcion[auxHmm]; (*HMM Calificado*)
  auxHmm
]
]
```



```
(*Llena las celdas de la correlacion de la matriz de Varianzas. matrix
solo tiene la diagonal con datos. numVar indica el num de variables
participantes*)
fillCeldasCorrelat[matrix_,numVar_]:=Block[{mat = matrix, i, j},
(*len=Length[mat[[1]]];*)
For[i = 0, i < numVar,
  For[j = i, j < numVar,
    Print["Valores: ",i," ",j];
    (*Se genera el valor entre el rango dado*)
    mat[[i,j]] = RandomReal[{(mat[[i,i])/(numVar + 1)),
      (mat[[i,i])/numVar}]);
    mat[[j,i]] = mat[[i,j]],
    j++],
  i++];
mat
]
```

```
(*Valida que la diagonal principal de la matriz de varianza, no
tenga un 0. En caso de encontrar un 0, lo cambia por 0.01. tmpClus,
es la lista de datos del cluster. numVar es el numero de variables*)
Clear[ValidaVariance];
ValidaVariance[tmpClus_,numVar_]:=Block[{matCov,j},
  matCov = Table[0,{numVar},{numVar}]; (*Variance Matrix*)
  For[j = 0, j < numVar,
    Print["Val Variance: ",Variance[tmpClus[[All,j]]]];

    If[Length[tmpClus] > 1, (*Calcula Varianza de los datos j*)
      (*If[Variance[tmpClus[[All,j]]] != 0.,*)
      If[Variance[tmpClus[[All,j]]] > 0.001,
        matCov[[j,j]] = Variance[tmpClus[[All,j]]],
        matCov[[j,j]] = 0.01
      ],
      (*Else Length[tmpClus] <= 1*)
      (*If[tmpClus[[1,j]] != 0., *Valor de dato != 0. *)
      If[tmpClus[[1,j]] > 0.001,
        matCov[[j,j]] = Abs[tmpClus[[1,j]]], (*la varianza es el mismo dato*)
        matCov[[j,j]] = 0.01
      ];
    ],
  j++];
matCov
]
```



```
(*Calcula la diagonal principal como la varianza propia de la
variable. Regresa la Matriz de Covarianza. Funcion que determina
las matrices de covarianza para cada Edo o Cluster dado.
Calculando la varianza de los datos de cada variable*)
Clear[FindCovarianceCluster];
FindCovarianceCluster[clusters_]:=Block[
{numClus,numVar,lMatsCov,tmpClus,i,mat},
  numClus = Length[clusters]; (*Num de Cluster*)
  numVar = Length[clusters[[1,1]]]; (*Num de Variables*)
  lMatsCov=Table[0,{numClus}]; (*Lista con la matrices*)
  For[i = 0, i < numClus,
    tmpClus = clusters[[i]]; (**)
    mat = ValidaVariance[tmpClus, numVar];
    lMatsCov[[i]] = fillCeldasCorrelat[mat, numVar],
    (*lMatsCov[[i]] = ValidaVariance[tmpClus, numVar],*)
    i++
  ];
  lMatsCov
]
```

```
(*Funcion que regresa las Medias de los datos de las variables
de los Clusters dados*)
Clear[FindMeanCluster];
FindMeanCluster[clusters_]:=Block[{numClus,numVar,tablaM,
tmpClus,i,j},
  numClus=Length[clusters]; (*Num de Clusters*)
  numVar=Length[clusters[[1,1]]]; (*Num de Variables*)
  tablaM=Table[0,{numClus}];
  For[i=0,i< numClus,
    tmpClus=clusters[[i]];
    tablaM[[i]]=Table[Mean[tmpClus[[All,k]]],{k,numVar}],
    i++
  ];
  (*Lista con las medias de los datos*)
  tablaM
]
```



```
(*Funcion que calcula el volumen (o hipervolumen). Es decir,
calcula la probabilidad de un conjunto de valores. Recibe el
vector de Means {3.1} de un Edo, y la Matriz de Covarianza
{{2.3}} de dicho Edo, una sublista de la lista de valores de
las variables a calcular {4.5,6.7,4.5} y un epsilon,0.10*)
Clear[FuncProbMult];
FuncProbMult[vMean_,matCovar_,lisVars_,epsi_]:=
Block[{len=Length[lisVars],nomVarlim,valLims,cade,j},

(Print["Entrando en FuncProbMult...: ",matCovar];*)

(*NIntegrate[PDF[MultinormalDistribution[vMean,matCovar],
  lValVars],listLim]*)
cade="NIntegrate[PDF[MultinormalDistribution["<>ToString[vMean]
  <>","<>ToString[matCovar]<>"],"<>ToString[lisVars]<>"],"";
nomVarlim=Map["a"<>ToString[#1]&,Range[len]];
valLims=listLimites[lisVars,epsi];
(Print["valLims: ",valLims];*)
For[j=0,j<len,
  ToExpression[nomVarlim[[j]]<> "="<>ToString[valLims[[j]]]];
  cade=cade<>ToString[ToExpression["a"<>ToString[j]]<>"],"",
  j++
];
cade=StringDrop[cade,-1]<>"]";

(Print["Por salir de FuncProbMult...: ",cade];*)

ToExpression[cade]
]
```



```
(•Parametros de un HMM:
hmdl [ 1 ] (•fitness•)
hmdl [ 2 ] (•size•)
hmdl [ 3 ] (•transitions•)
hmdl [ 4 ] (•vector de medias•)
hmdl [ 5 ] (•matriz de covarianza•)
hmdl [ 6 ] (•vector Pi•) *)

(•Regresa el Fitness de un HMM en particular•)
getFitnessHmm[HMM_]:=HMM[ [1,1] ]

(•Regresa el Tamaño de un HMM en particular•)
getSizeHmm[HMM_]:=HMM[ [2,1] ]

(•Regresa las Transiciones de un Edo en particular•)
getTransEdo[HMM_,numEdo_]:=HMM[ [3,1,numEdo] ]
(•Regresa Todas las Transiciones de un HMM dado•)
getAllTrans[HMM_]:=HMM[ [3,1] ]

(•Regresa el Vector de Medias de un HMM en particular•)
getMeansHmm[HMM_]:=HMM[ [4,1] ]
(•Regresa la Media de un Edo. en particular•)
getMeansEdo[HMM_,numEdo_]:=HMM[ [4,1,numEdo] ]

(•Regresa la Matriz de Covarianza de un HMM en particular•)
getCovarianzaHmm[HMM_]:=HMM[ [5,1] ]
(•Regresa la Matriz de Covarianza de un Edo. en particular•)
getCovarianzaEdo[HMM_,numEdo_]:=HMM[ [5,1,numEdo] ]

(•Regresa el Vector Pi de un HMM en particular•)
getPiHmm[HMM_]:=HMM[ [6,1] ]
(•Regresa el valor de Pi de un Edo. en particular•)
getPiEdo[HMM_,numEdo_]:=HMM[ [6,1,numEdo] ]
```



```
(*Se recibe la lista de transiciones del Edo, junto con el Num. de
Edo (o posicion) a modificar en la transicion. Una vez actualizada la
transicion, se mantiene la suma en 1*)
Clear[distrProbTrans];
distrProbTrans[transitionEdo_,newTransition_]:=Block[
{trans=transitionEdo,cont=0,increm,newTrans=newTransition},
Table[If[trans[[i]]!=0,cont++],{i,Length[trans]}];
increm=N[1/(cont+1)]; (*Nueva transicion*)
If[cont > 1,
Table[If[trans[[i]]!=0,
trans[[i]]=trans[[i]]-(trans[[i]]*increm/(1-increm)),
{i,Length[trans]}];
(*Edo a agregar transition*)
trans[[newTrans]]=increm +(increm*increm/(1-increm)),
(*else cont=1*)
Table[If[trans[[i]]!=0,trans[[i]]=increm,{i,Length[trans]}];
trans[[newTrans]]=increm;
];
Table[If[trans[[i]]==0.,trans[[i]]=0],{i,Length[trans]}];
trans
]
]
```

```
(*Llama a la funcion de copiar un cromosoma*)
Clear[Copy];
Copy[chromoPop_pop, opts___] :=
Module[{}, (*pop[chromoPop//First]*)
pop[Selections[chromoPop,opts]//First]
]
]
```

```
(*Llama a la funcion que Agrega una nueva transicion*)
Clear[AddTransition];
AddTransition[chromoPop_pop,opts___]:=Module[{},
pop[AddTrans[Selections[chromoPop,opts]//First]]
]
]
```

```
(*Primero se determina en el edo seleccionado de manera aleatoria,
cuales posiciones estan marcadas con 0 (sin transiciones)*)
Clear[AddTrans];
AddTrans[HMM_] :=
Block[{auxHmm=HMM,newTrans,numEdo,addTrans},
numEdo=RandomInteger[{1,getSizeHmm[auxHmm]}];
newTrans=Position[getTransEdo[auxHmm,numEdo],0];
If[newTrans==={},
auxHmm,
auxHmm[[3,1,numEdo]]=AddTransition[auxHmm,numEdo,newTrans];
];
auxHmm
]
]
```

```
(*Se recibe un HMM, un Num de Edo, y la lista de Transiciones
Disponibles a modificar.*)
AddTransition[Hmm_,numEdo_,transDispo_]:=
Block[{auxHmm=Hmm,newTrans,cont=0,transEdo,incem},
  transEdo = getTransEdo[auxHmm,numEdo]; (*Transiciones del Edo*)
  If[Length[transDispo] > 1, (*Mas de una espacio*)
    newTrans = transDispo[[RandomInteger[{1,Length[transDispo]}]]];
    newTrans = newTrans[[1]],
    newTrans = transDispo[[1,1]]
  ];
  (*Se pasa el Num de Edo y la posicion a insertar el valor*)
  distrProbTrans[transEdo,newTrans]
]
```

```
(*Funcion que inserta un 0 al final de las transiciones, y agrega
una nueva lista, al final de las Trans., con 0s. Llamado por la funcion
AddState*)
Clear[insNewTransEdos];
insNewTransEdos[matTransiciones_]:=
Block[{allTrans=matTransiciones,len},
  len=Length[allTrans];
  (*Se agrega al final de cada lista de los Edos,la nueva transicion*)
  allTrans=Table[Insert[allTrans[[i]],0,-1],{i,len}]; (*Se agrega columna*)
  (*Se agrega una lista de 0s al final de la matriz*)
  allTrans=Insert[allTrans,Table[0,{len+1}],-1];
  (*Se insertan valores y se normaliza la transicion*)
  allTrans[[len+1]]=Transitions[len+1];
  allTrans
]
```

```
(*Llama a la funcion de borrar una transición del HMM*)
Clear[DeleteTransition];
DeleteTransition[chromoPop_pop,opts___]:=Module[{} ,
  pop[DeleteTransition[Selections[chromoPop,opts]//First]]
  (*pop[delTransition[chromoPop//First]]*)
]
```




```
(*Se recibe el HMM y se determina el num. de Edo a eliminar
su transicion. Se verifica que el Edo tenga transiciones para
eliminar, > 1*)
DeleteTransition[HMM_]:=
Block[{auxHmm=HMM,delTrans,Alltrans,edo,sizel=getSizeHmm[HMM]},
  delTrans=RandomInteger[{1,getSizeHmm[auxHmm]}];
  Alltrans=getAllTrans[auxHmm];(*auxHmm[[3,1]]; Todas las
                                transiciones del HMM*)
  If[Count[Alltrans[[delTrans]],0]!=(sizel-1),
    edo=RandomInteger[{1,sizel}];
    Alltrans[[delTrans,edo]]=0;
    (*Alltrans[[delTrans]]=normalTrans[Alltrans[[delTrans]]];*)
    (*auxHmm[[3,1]]=Table[normalTrans[Alltrans[[i]]],{i,sizel}];*)
    Alltrans[[delTrans]]=normalTrans[Alltrans[[delTrans]]];
  (*Print["No. Edo: ",delTrans," trans: ",Alltrans[[delTrans]]];*)
  auxHmm[[3,1]] = Alltrans;
  ];
  auxHmm
]
```

```
(*Inicia la mutacion de parametros. Se recibe a las cromosomas,
se selecciona la mejor y se realiza la mutacion*)
Clear[MutParameters];
MutParameters[chromoPop_pop,opts___]:=Block[{},
  pop[MutParameters[Selections[chromoPop,opts]//First,opts]]
]
```

```
iniOptionParameters[paraMean_,stepM_,varCov_]:=Block[{},
  Options[MutParameters]={
    pm -> paraMean,
    sSizeMean -> stepM,
    sSizeCov -> varCov} ]
```

```
(*Toma, en caso de ser necesario, los valores por default.
Incrementa el valor de las medias del Edo correspondiente y
calcula tambien la matriz de covarianza del mismo Edo, en
base a las medias y al incremento de la covarianza*)
MutParameters[HMM_,opts___]:=
Block[{auxHmm=HMM,mMean,sSM,sSV,len,hmmMeans,j,
k,lenVars,matCovar,valVar},

(*Print["Entrando a MutParameters"];*)

mMean=pm/.{opts}/.Options[MutParameters];
sSM=sSizeMean/.{opts}/.Options[MutParameters];(*Step Size Mean*)
sSV=sSizeCov/.{opts}/.Options[MutParameters];(*Step Size Variance*)
len=getSizeHmm[auxHmm];(*Num de Estados del HMM*)
hmmMeans=getMeansHmm[auxHmm];(*HMM[[4,1]]*)
lenVars=Length[hmmMeans[[1]]];(*Num de Variables aleatorias*)
For[j = 0,j < len,
  (*mMean valor tope para entrar o no a modificar parametros
  en cada estado del HMM, se compara contra un valor generado
  aleatoriamente entre 0 y 1*)
  If[RandomReal[{0,1}] < mMean,
    matCovar=getCovarianzaEdo[auxHmm,j];(*HMM[[5,1,numEdo]]*)
    For[k=0,k < lenVars,
      hmmMeans[[j,k]]=hmmMeans[[j,k]]+
        RandomReal[NormalDistribution[0,sSM]];
      If[(valVar = matCovar[[k,k]]+
        RandomReal[NormalDistribution[0,sSV]]) != 0.,
        valVar = Abs[valVar],
        valVar = 0.01 ];
      matCovar[[k,k]] = valVar,
      k++];
      auxHmm[[4,1]] = hmmMeans;(*Asignacion Means Mutado*)
  (*Print[matCovar];*)
  auxHmm[[5,1,j]] = matCovar(*Matriz Varianza Mutada*)
  ],(*If mMean*)
  j++];
auxHmm
]
```

```
Clear[MutTransitions];
MutTransitions[chromoPop_pop,opts___]:=Block[{} ,
  (*pop[mutTransitions[chromoPop//First,opts]]*)
  pop[MutTransitions[Selections[chromoPop,opts]//First,opts]]
]
```

```
Options[MutTransitions]={ stepSize -> 0.7 };
```

```
(*Selecciona aleatoriamente el Edo, calcula el incremento y
realiza los incrementos a todos los valores de la lista.
Despues normaliza los valores a 1, y asigna el valor actualizado
al HMM*)
MutTransitions[HMM_,opts___]:=
Block[{auxHmm=HMM,transEdo,edoMut,sSize,inc,nEdos},
  nEdos = getSizeHmm[auxHmm];
  sSize = stepSize/.{opts}/.Options[MutTransitions];
  edoMut = RandomInteger[{1, nEdos}];
  inc = calcIncrem[sSize];
  transEdo = getTransEdo[auxHmm, edoMut];(*Obtiene las transiciones*)
  transEdo = Table[Abs[transEdo[[i]] + inc], {i, nEdos}];
  (*Asigna lista actualizada*)
  auxHmm[[3, 1, edoMut]] = normalTrans[transEdo];
  auxHmm
]
```

```
(*Llama a la funcion que elimina un Estado*)
Clear[DeleteState];
DeleteState[chromoPop_pop,opts___]:=Module[{auxHmm},
  auxHmm = Selections[chromoPop,opts]//First;
  If[getSizeHmm[auxHmm] > (min/.{opts}/.Options[Evolution]),
    auxHmm = DeleteState[auxHmm] ];
  (*pop[DeleteState[Selections[chromoPop,opts]//First]]*)
  pop[auxHmm]
]
```

```
(*Modulo que se encarga de eliminar un edo. del HMM dado.
Elimina las transiciones, reduce los edos., etc. Es decir,
actualiza el HMM a un edo menos*)
DeleteState[HMM_,opts___]:=
Block[{auxHmm=HMM,randEdo,nEdos,allTrans,i,timeSeries,nClusters},
  nEdos = getSizeHmm[auxHmm];
  If[nEdos > 2, (*Solo para HMM con mas de 2 Edos*)
    randEdo = RandomInteger[{1, nEdos}];
    nEdos = nEdos - 1; (*Se considera Eliminado un Edo*)
    timeSeries = tSeries/.{opts}/.Options[initChrome];
    (*Borra Transicion Edo*)
    allTrans = Delete[getAllTrans[auxHmm],randEdo];
    Table[allTrans[[i]] = Delete[allTrans[[i]], randEdo],
      {i, nEdos}];(*Transicion eliminada*)
    auxHmm[[3,1]] = Table[normalTrans[allTrans[[i]]],{i,nEdos}];(*Transiciones Normalizada
    auxHmm[[2,1]] = nEdos;(*Num Edos actualizado*)
    nClusters = FindClusters[timeSeries, nEdos];
    auxHmm[[4,1]] = FindMeanCluster[nClusters];
    auxHmm[[5,1]] = FindCovarianceCluster[nClusters];
    auxHmm[[6,1]] = normalTrans[Delete[getPiHmm[auxHmm],randEdo]];
  ];
  auxHmm
]
```

```
(•Llama a la funcion Agregar un Nuevo Estado•)
Clear[AddState];
AddState[chromoPop_pop,opts___]:=Module[{} ,
  pop[AddState[Selections[chromoPop,opts]//First,opts]]
]
```

```
(•Funcion que agrega un nuevo Edo., al HMM. Ejecutando lo necesario
para dejar al HMM de manera correcta. Se agregan: transiciones, medias,
matriz de covarianza, etc. Se retorna el nuevo HMM•)
AddState[HMM_,opts___]:=
Block[{auxHmm=HMM,nClusters,nEdos,timeSeries},
  nEdos = getSizeHmm[auxHmm] + 1; (*se suma el Edo al HMM*)
  timeSeries = tSeries/.{opts}/.Options[initChrome];
  nClusters = FindClusters[timeSeries,nEdos];
  auxHmm[[2,1]] = nEdos;
  auxHmm[[3,1]] = insNewTransEdos[getAllTrans[auxHmm]];
  auxHmm[[4,1]] = FindMeanCluster[nClusters];
  auxHmm[[5,1]] = FindCovarianceCluster[nClusters];
  auxHmm[[6,1]] = Table[N[1/nEdos],{nEdos}];
  auxHmm
]
```

```
(*****
FUNCIONES DECLARADAS O QUE SE HAYAN EN EVOLVICA
*****)
```

```
Options[selectBest] = { Best -> 1 };
```

```
selectBest[fitnesses_, opts___] :=
Module[{bestCount, fitsAndIndsSorted},

  bestCount = Best /. {opts} /. Options[selectBest];

  fitsAndIndsSorted =
    Reverse[Sort[MapIndexed[ {#1, First[#2]} &, fitnesses]]];

  Last /@
  If[IntegerQ[bestCount],
    Take[fitsAndIndsSorted, bestCount],
    Take[fitsAndIndsSorted,
      Max[1, Floor[bestCount Length[fitnesses]]]]
  ]
]
```

```
(•Regresa un operador, de evolvida de manera aleatoria•)
selectFitProp[fitnesses_, opts___] :=
Module[{cumFits, fits, r, m},
  fits =
  If[Negative[m = Min[fitnesses]], fitnesses -m, fitnesses];
  cumFits = FoldList[Plus, 0, fits] // Rest;
  r = Random[] Last[cumFits];
  Position[cumFits, _?(# >= r &), {1}, 1][[1,1]]
]
```



```
Options[selectTournament]={
    TournamentCompetitors -> 5,
    Best -> 1 };

selectTournament[fitnesses_,opts___]:=
Module[{howManyCompetitors,fitsAndInds,competitors,wins,fit,
    scoresAndIndsSorted},

    howManyCompetitors=(TournamentCompetitors/.{opts}/.Options[
        selectTournament]);

    fitsAndInds=MapIndexed[{{#1,First[#2]}}&,fitnesses];

    competitors=
    Map[
        randomSubset[fitsAndInds,howManyCompetitors,{#}]&,
        fitsAndInds];

    wins=MapThread[(fit=First[#1];
        Count[Map[(fit>First[#])&,#2],True])&,
        {fitsAndInds,competitors}];

    scoresAndIndsSorted=Map[Last,Sort[MapIndexed[{{#1,First[#2]}}&,
        wins]]];

    Sort[Take[
        scoresAndIndsSorted,-Best/.{opts}/.Options
            [selectTournament]]]
]
```

```
selectRankBased[fitnesses_,opts___]:=
Module[{fitsAndIndsSorted,index},

    fitsAndIndsSorted=
    Sort[MapIndexed[{{#1,First[#2]}}&,fitnesses]];

    index=selectFitProp[Range[Length[fitnesses]]];

    fitsAndIndsSorted[[index]]//Last
]
```

```
Options[selectElite] = { Elite -> 1 };
```

```
selectElite[fitnesses_, opts___] :=
Module[{bestCount, fitsAndIndsSorted},

  bestCount=Elite/.{opts}/.Options[selectElite];

  fitsAndIndsSorted =
    Reverse[Sort[MapIndexed[{#1, First[#2]} &, fitnesses]]];

  selectRandom[
    Last /@
    If[IntegerQ[bestCount],
      Take[fitsAndIndsSorted, bestCount],
      Take[fitsAndIndsSorted,
        Max[1, Floor[bestCount Length[fitnesses]]]]
    ]
  ]
]
```

```
selectFitProp[fitnesses_, opts___] :=
Module[{cumFits, fits, r, m},
  fits=If[Negative[m=Min[fitnesses]], fitnesses -m, fitnesses];
  cumFits=FoldList[Plus, 0, fits]//Rest;
  r=Random[] Last[cumFits];
  Position[cumFits, _?(# >= r &), {1}, 1][[1,1]]
]
```

```
(*Extrae en una Lista, todos los fitnesses de los HMMs*)
extractQuals[population_pop] :=
  List @@ Map[First[First[#]] &, population]
```

```
prettyPrintFitnesses[ep_pop] :=
Module[{fitnesses},
  fitnesses = Sort[Cases[ep, _q, Infinity]/.{q[x_] :> x}];
  Print["Fitnesses: ", Sort[fitnesses]];
]
```

```
Options[Selections] = { SelectionMode -> FITPROP };
```



```
(*Esquema de selección de los individuos mas aptos*)
Selections[population_pop,howMany_Integer:1,
  opts___] := Module[{},
  Switch[
    SelectionMode/.{opts}/.Options[Selections],

    (* BEST,Take[Sort[population,getQ[#1]>getQ[#2]&],howMany],*)
    BEST,
      Take[population//Sort//Reverse,howMany],

    RANDOM,
      pop @@
      Table[population[[RandomInteger[{1,Length[population]}]]],
        {howMany}],

    FITPROP,
      Table[population[[
        selectFitProp[extractQuals[population],opts]
      ]],{howMany}],

    RANKBASED,
      Table[population[[
        selectRankBased[extractQuals[population],opts]
      ]],{howMany}],

    ELITE,
      Table[population[[
        selectElite[extractQuals[population],opts,
          Elite -> 1.]
      ]],{howMany}],

    TOURNAMENT,
      population[[
        selectTournament[extractQuals[population],opts,
          Best -> howMany
        ]
      ]
    ]
  ];
```



```
(*Opciones por defecto de la Evolucion*)
Options[Evolution] = {
  InitialPopulation -> {},
  SelectionMode -> FITPROP,

  Operators->({MutParameters , 0.3},
              {Copy , 0.0}, {MutTransitions , 0.4},
              {DeleteTransition , 0.4},
              {AddTransition , 0.5},
              {AddState , 0.7}, {DeleteState , 0.4}},

  ExprMaxDepth -> 2,
  ExprPatterns -> {},
  ExprAtoms -> {},
  min -> 2,
  max -> 10
};
```

```
(*Funcion evolucion de Evolvica*)
Evolution[
  EPHMM[
    m_?IntegerQ, (* parents *)
    s_?AtomQ, (* PLUS or COMMA *)
    l_?IntegerQ, (* children *)
    g_?IntegerQ, (* generations *)
    opts___]i_1 (* options *) (* independent runs *)
]:= Module[{initialParents, evalFct, parents, children,
            selPool,selectOperator, opsAndProbs, gen},

  Print["EP experiment ..."];

  (* Define an operator selection function *)
  opsAndProbs = Operators /. {opts} /. Options[Evolution];

  selectOperator := First[Transpose[opsAndProbs]][[
    selectFitProp[Last[Transpose[opsAndProbs]]]]];

  If[{(initialParents = {InitialPopulation /. {opts}
    /. Options[Evolution]}) == {}},

    initialParents=GenInitPop[m,min/.{opts}/.Options[Evolution],
    max/.{opts}/.Options[Evolution]]
  ];

  (* Get the evaluation function *)
  evalFct = EvaluationFunction/.{opts}/.Options[Evaluation];

  Print["Generation ",gen = 0];
  initialParents=Map[Evaluation[#,EvaluationFunction :> evalFct]&,
    initialParents];

  Table[
    (* Iterate for g generations *)
    NestList[
      ( Print["Generation ", ++gen];
```



```

parents = #;
Print[parents[[1]]];

(*Print["Calificacion: ", parents[[1,1,1]]];*)

If[parents[[1,1,1]] > 0.97, (*Condicion de SALIDA*)
  Print["SALIR..."]; Return[parents[[1,1,1]]
];

prettyPrintFitnesses[parents];

children = pop[];

While[Length[children] < 1,
  children = children -Join-
    selectOperator[parents,opts]
];

children =
  Map[Evaluation[#,EvaluationFunction :> evalFct]&, children];

selPool = Switch[s,
  PLUS,Join[parents,children],COMMA,children];

Selections[selPool, m, SelectionMode -> BEST]

) &, initialParents, g
], {i}
] (*Table*)
] (*Module*)

```

```

iniOptionChrome[seTime_, epsi_] := Block[{},
  Options[initChrome] = { tSeries -> seTime,
    epsilon -> epsi } ]

```

```

(*Inicia las Chromosomes de manera aleatoria, entre un rango
de 2 a N numero de Edos.*)
Clear[initChrome];
initChrome[EdosMin_:2,EdosMax_?IntegerQ,opts___] :=
Block[{nEdos,nClusters,timeSeries},
nEdos=RandomInteger[{EdosMin,EdosMax}];
timeSeries = tSeries/.{opts}/.Options[initChrome];
nClusters=FindClusters[timeSeries,nEdos];
HMM[q[undef],
size[nEdos],
transitions[Table[Transitions[nEdos],{nEdos}]],
medias[FindMeanCluster[nClusters]],
covarianza[FindCovarianceCluster[nClusters]],
pi[Table[N[1/nEdos],{nEdos}]]
]
];

```

```

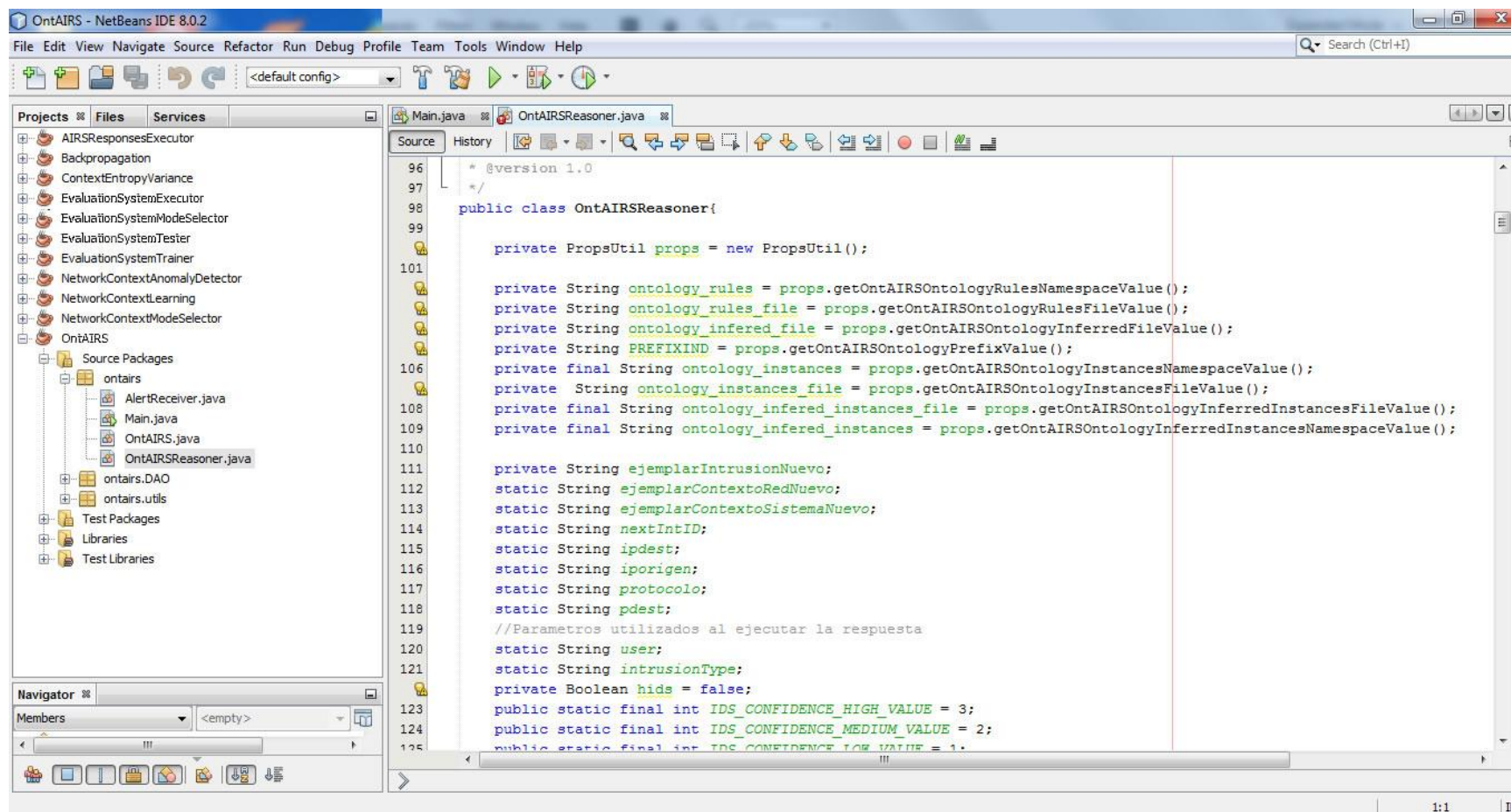
Clear[GenInitPop];
GenInitPop[nPop_,Min_,Max_] :=
Apply[pop,Table[initChrome[Min,Max], {nPop}]];

```



2. Entorno de Desarrollo

Figura 28. Entorno de Desarrollo del AIRS Proactivo Componente Principal



Fuente. Captura de Pantalla de Entorno de Desarrollo

Tabla 21. Tráfico de datos de entrada de la red de la municipalidad distrital de Chongoyape (Kbit/segundo)

Día Hora	1	2	3	4	5	6	7
1	11.916	12.6	29.452	13.097	27.545	6.774	176.019
2	9.178	11.191	28.201	7.09	26.269	16.636	164.117
3	6.012	7.078	25.493	10.714	24.656	7.57	2460.41
4	15.993	7.18	25.046	4.619	27.205	6.085	1147.19
5	22.007	7.874	24.414	6.315	25.807	7.546	76.099
6	7.278	8.121	24.29	6.242	8.111	12.943	82.741
7	21.624	123.741	26.453	26.489	15.222	8.374	75.737
8	381.954	489.682	214.319	305.644	297.09	13.383	76.977
9	1599.59	1367.38	1064.69	1112.62	1537.31	40.597	111.361
10	3316.84	1206.21	1946.45	1037.68	868.032	114.597	94.568
11	1803.96	1048.31	1829.29	842.686	1677.72	184.489	96.457
12	1290.61	2067.52	2531.96	1712.81	3457.12	131.861	139.657
13	1385.08	1262.21	2991.55	1960.48	1392.12	1801.5	315.029
14	1123.56	1200.06	1886.84	2419.93	1277.62	625.129	236.728
15	750.931	772.793	1292	1047.85	741.622	776.635	175.68
16	1778.59	558.78	1030.36	1067.28	1840.53	499.832	245.655
17	1724.14	618.056	1057.97	989.765	923.752	320.427	174.335
18	1731.92	829.955	1400.9	3475.78	765.924	280.397	84.096
19	1328.41	5618.73	602.418	1324.86	1188.02	37.689	16.734
20	660.876	678.117	544.322	439.822	659.746	9.238	53.448
21	241.328	279.244	2090.78	625.583	360.39	163.621	248.861
22	63.9	107.279	120.732	149.583	57.075	210.364	23.997
23	33.979	37.668	16.673	44.576	10.228	166.412	6.872
24	16.039	27.882	9.097	31.896	7.56	108.535	3.934



Tabla 22. Tráfico de datos de salida de la red de la municipalidad distrital de Chongoyape (Kbit/segundo)

Día Hora	1	2	3	4	5	6	7
1	188.61	140.206	120.662	213.639	81.459	30.668	105.058
2	208.538	84.203	155.079	73.878	65.982	45.344	100.359
3	148.322	60.269	54.783	191.047	79.365	23.302	216.415
4	128.886	48.215	52.256	27.067	73.821	13.568	190.596
5	131.952	36.551	36.149	68.104	89.164	31.278	157.742
6	147.386	89.598	64.702	53.958	51.396	15.654	143.298
7	127.47	97.088	112.284	94.108	114.521	25.493	150.603
8	204.024	188.484	99.55	98.893	104.963	46.724	163.737
9	442.06	272.698	287.142	276.014	249.487	36.273	164.888
10	711.892	566.965	484.124	371.257	381.957	64.656	150.167
11	578.611	635.393	779.745	557.307	586.978	69.819	138.182
12	584.167	580.844	1581.36	519.888	646.537	84.664	173.155
13	589.181	455.343	669.146	607.856	453.7	156.384	244.193
14	487.042	497.216	618.001	611.396	200.347	118.038	163.196
15	371.549	407.399	790.635	531.858	173.294	157.898	878.878
16	509.003	513.217	728.849	415.401	358.016	118.473	910.102
17	586.864	461.462	746.352	428.925	310.92	97.919	279.089
18	585.3	439.972	797.397	543.634	269.759	158.609	169.419
19	615.943	745.652	403.389	478.228	208.194	37.804	121.502
20	429.878	493.694	316.742	294.422	424.699	59.738	213.568
21	381.275	279.284	313.224	333.095	68.061	115.213	144.371
22	216.703	241.115	436.07	207.763	46.385	145.078	91.172
23	299.08	200.035	134.844	275.598	54.547	146.634	68.02
24	195.913	170.022	141.542	188.128	51.794	160.616	41.763

