UNIVERSIDAD DE MURCIA

Facultad de Informática

DEFINICIÓN DE UN SIMULADOR PARA EL ANÁLISIS DE TÉCNICAS DE DETECCIÓN DE INTRUSIONES BASADAS EN MODELOS DE GRAFOS DE ATAQUE

PROYECTO FIN DE CARRERA

Presentada por:

Juan José Andreu Blázquez

Supervisada por:

D. Manuel Gil Pérez

Dr. Gregorio Martínez Pérez

Murcia, Septiembre de 2015

Índice de contenidos

Índice de figuras

Resumen

En el campo de la Informática, la seguridad es un especto fundamental. Sin el establecimiento de una política de seguridad adecuada, una organización puede verse seriamente amenazada. En la seguridad de la información, es imprescindible analizar los riesgos de ataque a los que se ve expuesto un sistema o red. En el área de la seguridad que se encarga de la detección de ataques, una de las técnicas que permiten llevar a cabo su análisis es el de los Grafos de Ataques.

Los grafos de ataques son una herramienta que permite utilizar información sobre la estructura de un sistema, y los puntos débiles que contiene, para diseñar un grafo mediante el cual identificar el camino que seguiría un atacante en su afán de comprometer nuestro sistema. Suponen un modelo robusto sobre el que analizar la seguridad.

En la actualidad existen una gran cantidad de variados modelos de grafos de ataques que se pueden utilizar para analizar la seguridad. Estos modelos utilizan diferentes enfoques sobre el concepto de grafo de ataques, presentando soluciones alternativas. Con la gran cantidad de soluciones propuestas en la literatura, resulta complicado decantarse por uno u otro a la hora de elegir el más adecuado para el análisis de las características de una red concreta.

Por ello, y con el objetivo de resolver ese problema, en este trabajo se presenta una herramienta de software que sirve como simulador de diferentes modelos de grafos de ataques. La herramienta pretende servir como un elemento de ayuda a la hora de desplegar nuevos modelos. También tiene como objetivo analizar los ya existentes, y dar soporte a la hora de comparar diferentes modelos, e incluso auxiliar en la tarea de identificar aquellos aspectos que un modelo que puedan no estar bien definidos, durante su fase de desarrollo.

El simulador presentado se apoya en un diseño flexible, definiendo las estructuras básicas y las interfaces de comunicación generales, que permitirá la inclusión de modelos de forma bastante sencilla, y habilitará el estudio del comportamiento de diferentes modelos de grafos de ataques bajo circunstancias parecidas.

1. Introducción

La seguridad, en su definición más general, se define como el «*estado de bienestar que percibe y disfruta el ser humano*». Cuando se habla de seguridad en un ámbito más científico, se define como una «*ciencia interdisciplinaria, encargada de evaluar, estudiar y gestionar los riesgos a que se encuentra sometido una persona, un bien o el ambiente*». Finalmente, en el caso de las tecnologías de la información, la seguridad suele definirse como el conjunto de medidas preventivas y reactivas de las organizaciones y de los sistemas tecnológicos que permiten resguardar y proteger la información, para evitar riesgos o daños. Un sistema es seguro si está libre de daños, ya sean internos o externos, intencionados o no. Conseguir que un sistema sea completamente seguro es algo utópico, puesto que es imposible controlar todas las posibles contingencias que ponen en riesgo un sistema de información. El proceso de aplicar medidas de seguridad en un sistema de información debe esmerarse en alcanzar ciertas características sobre la información en él contenida. Estas características suelen conocersse como la triada *CIA*, cuyas siglas se corresponden con *Confidentiality (Confidencialidad), Integrity (Integridad), Availability (Disponibilidad)* [1]:

- Confidencialidad: Es la característica mediante la que se reserva el acceso a la información solamente a aquellas entidades con autorización para ello.
- Integridad: Esta característica debe garantizar que la información solo es modificada por una entidad con la debida autorización para ello. Así como evitar la alteración de ésta por entidades no autorizas, u otras causas.
- Disponibilidad: Es la característica mediante la que se garantiza que la información debe estar disponible para su uso siempre que sea necesario. Los sistemas deben ser accesibles y funcionar correctamente.

En la actualidad, cualquier organización dispone de una red de dispositivos informáticos que puede oscilar desde una decena hasta varios miles de dispositivos, en su mayoría conectados de forma permanente a Internet. Estas redes de ordenadores necesitan ser protegidas para garantizar el buen funcionamiento de dichas organizaciones, y evitar daños derivados de ataques o fallos en los sistemas de información de dichas organizaciones.

El *Análisis y gestión de riesgos* es una disciplina utilizada para estudiar y estimar la probabilidad de que tenga lugar un incidente que afecte a los sistemas de la red de una organización, y el impacto y alcance de las consecuencias de dicho suceso. Esto sirve para elaborar y poner en marcha medidas preventivas y/o reactivas que eviten o minimicen el impacto de dichos sucesos sobre los sistemas y redes de la organización.

Los *grafos de ataques* son una herramienta importante para analizar vulnerabilidades y riesgos de seguridad en redes de comunicaciones. Dichos grafos se utilizan para determinar si ciertos estados objetivo pueden ser alcanzados por atacantes intentando penetrar redes de ordenadores desde unos estados iniciales. Son grafos en los que el nodo inicial representa a un atacante en un punto de la red. Los nodos y aristas representan acciones llevadas a cabo por el atacante y cambios en el estado de la red causados por esas acciones. Dichas acciones suelen consistir en explotar y aprovecharse de vulnerabilidades presentes en el software o los protocolos utilizados en la red. En grandes grafos de ataques puede ser necesario comprometer diversos sistemas de la red para utilizarlos como puntos intermedios mediante los que se alcanza el objetivo. Un grafo de ataques completo contendrá todas las secuencias de acciones posibles para un atacante, que finalmente le llevan a obtener los privilegios deseados en el objetivo[2]. En la figura 1 se puede observar un ejemplo de grafo de ataque que permitiría a un atacante abrir una caja fuerte.

En la actualidad existen diversos tipos y enfoques de grafos de ataques que se pueden utilizar para modelar y analizar la seguridad de una red. Sin embargo, no resulta fácil decantarse por un tipo concreto para un escenario real, ya que muchos de dichos enfoques solamente se definen teóricamente o con ejemplos muy simples, mientras que otros utilizan diferentes métricas y a priori resulta difícil saber cuál es más adecuado. Otro problema a la hora de realizar un análisis mediante grafos de ataques es la escalabilidad. Cuando se pretende ser capaz de analizar un escenario en el que una red con varias decenas de elementos que están expuestos cada uno a varias vulnerabilidades potenciales, el número de nodos de los grafos de ataques resultantes crece de manera exponencial, y dichos grafos adquieren un tamaño inmanejable para una persona. Sin embargo, la mayoría de los modelos computacionales propuestos no han sido probados en escenarios de redes de más de 20 hosts, cuando en la práctica sería interesante poder analizar redes de organizaciones con varios miles de elementos. Además de lo anterior, también se presenta un problema adicional, puesto que

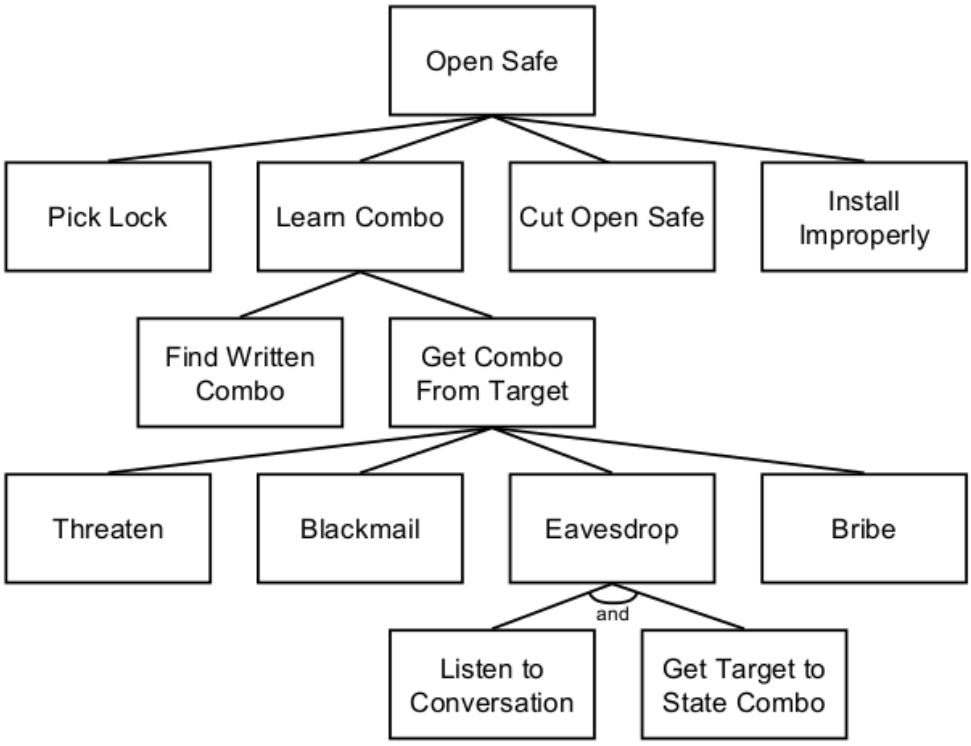

Figura 1: Grafo de ataques de una caja fuerte [3]

muchos de los grafos de ataques previamente propuestos, utilizan sus propias estructuras de representación de la información de entrada para dichos grafos, y al mismo tiempo para la salida generada.

Por estos motivos, el uso en la práctica de dichos modelos se convierte en una tarea complicada y llena de incertidumbre, tanto a la hora de elegir el modelo a utilizar, como a la hora de prever si dichos modelos podrán utilizarse en redes de tamaños superiores a alguna decena de hosts, y al mismo tiempo sin que exista una manera viable de conocer cómo de buenos serán los resultados.

Teniendo en cuenta los problemas existentes en los modelos de grafos de ataques en la actualidad (gran variedad de enfoques diferentes, dificultad para compararlos a priori, definiciones o ejemplos puramente teóricos en algunos de ellos, escalabilidad), en este trabajo se presenta un componente software que tiene como objetivo servir de ayuda en el despliegue de modelos de grafos de ataques. Dicho componente actúa como simulador para poder analizar

escenarios de redes utilizando varios modelos de grafos de ataques diferentes, de modo que pueda servir para ayudar a comparar técnicas de grafos de ataques bajo características similares, ya sean nuevos enfoques u otros ya existentes. Al mismo tiempo tiene como objetivo permitir identificar aquellos aspectos de modelos que no se hayan definido completamente y que deban ser refinados para poder ser utilizados en escenarios realistas. Para la consecución de estos objetivos y facilitar el uso de diversos modelos de grafos de ataques, se define una interfaz sencilla que cualquier modelo puede implemetar, habilitándolo para el uso en el simulador. Los modelos que implementen dicha interfaz, serán integrados en el simulador, y podrán ser probados junto a los demás en diferentes escenarios.

El simulador presentado en este trabajo es el primer resultado de un proyecto en común de un alumno de máster de la Universidad de Verona junto con otro de la Universidad de Murcia, y sus tutores. El presente proyecto ha dado lugar a un artículo presentado y aceptado en las *I Jornadas Nacionales de Investigación en Ciberseguridad (JNIC)*, bajo el título *Multigraph Project: First Steps towards the Definition of a Multiple Attack Graph Model Simulator*. Dichas jornadas tienen lugar los días 14, 15 y 16 de septiembre de 2015.

Este documento está dividido en secciones. La primera sección la comprende la presente introducción. La segunda de ellas muestra el objetivo que se quiere alcanzar con el desarrollo de este simulador, y el estado del arte de las soluciones actuales. La tercera sección muestra todo el diseño de la herramienta, su implementación y los modelos incluidos a modo de prototipo. En la cuarta sección se indican las conclusiones, y las vías futuras de desarrollo de este simulador.

2. Análisis de objetivos y metodología

En esta sección se exponen los objetivos que se han tratado de alcanzar con el desarrollo de este proyecto fin de carrera. Al mismo tiempo, se exponen las herramientas, técnicas y modelos analizados y estudiados y los finalmente utilizados para la consecución de dichos objetivos.

El propósito principal de este trabajo es sentar las bases de una herramienta de simulación de grafos de ataques en redes, para ello se cuenta con los siguientes objetivos:

- Observar el desempeño de diferentes modelos de grafos de ataques sobre un mismo escenario de red.
- Comparar el resultado de diferentes modelos entre sí, y poder establecer cuáles funcionan mejor o peor ante ciertas características.
- Identificar aquellos aspectos poco definidos de modelos de grafos de ataques que supongan un impedimento para su uso en escenarios reales.

En primer lugar, se explicarán los conceptos básicos de los grafos de ataques, además se analizará en detalle la situación actual de dicha técnica en la literatura. Esto servirá para conocer cuáles son las diferencias que se suelen encontrar entre diferentes modelos de grafos de ataque (2.1), y qué retos se les presentan, así como cuestiones aún por resolver (2.1.1). Después se pasará a definir y detallar el análisis de vulnerabilidades (2.2), y se mostrarán las bases de datos públicas de vulnerabilidades más importantes (2.2.1). Dichas bases de datos se han estudiado para conocer en profundidad la clasificación que se lleva a cabo de las vulnerabilidades publicadas. Por último, se definirán los eventos, y se detallará la categorización de eventos que podremos usar en la arquitectura (2.3).

2.1. Grafos de ataque

En este trabajo se ha realizado un simulador del comportamiento de diferentes modelos de grafos de ataques en redes, para poder aplicarlos y compararlos en la práctica. Por lo tanto, el

primer paso es conocer qué son los grafos de ataque, y cómo se aplican. Se entiende como grafo de ataque, a un grafo en el que se modelan y separan los pasos que se tienen que dar mediante objetivos individuales sencillos, hasta lograr un objetivo final más complejo, en el que el atacante obtiene lo que deseaba. Típicamente tienen una estructura de árbol, con un nodo raíz que es el objetivo final, y nodos hoja desde los que es susceptible de comenzar el ataque. Los nodos y aristas intermedios representan las acciones individuales del atacante y los cambios producidos por estas[2]. En los nodos y aristas se incluyen propiedades que pueden utilizarse durante el análisis de dichos grafos, dichas propiedades pueden ser: la probabilidad de que se lleve a cabo ese paso del ataque, el coste que tendría para el atacante, si necesita herramientas específicas para llevarse a cabo, si es legal, e incluso posibles contramedidas para evitarlo y el coste de implementarlas. Al mismo tiempo, varios grafos de ataques pueden unirse formando uno más grande que es capaz de representar ataques en varios sistemas a la vez, o ataques más complejos. En la figura 2 se puede observar un grafo de ataques con forma de árbol para comprometer el buen funcionamiento de un sistema SCADA.

Los grafos de ataques surgen como una herramienta que permite describir y analizar cómo un atacante puede penetrar en una red y comprometer su seguridad, de forma que se puedan estudiar contramedidas para evitar dichos ataques, conocer cuáles son los elementos más susceptibles de ser atacados, establecer adecuados sistemas de detección de dichos ataques o hacer un análisis forense una vez se ha producido un ataque.

En su presentación de 1999[3], Schneier otorgaba una alta cantidad de posibilidades y beneficios al uso de los grafos de ataque, entre las que destacaban:

- Determinar si un sistema es vulnerable ante un ataque.
- Obtener una lista de cualidades que se asumen como ciertas en el sistema (y pueden no serlo), qué pasa si dichas asunciones se cumplen.
- Determinar el impacto de una modificación en un sistema.
- Comparar y ordenar diferentes ataques en base a su probabilidad de éxito.

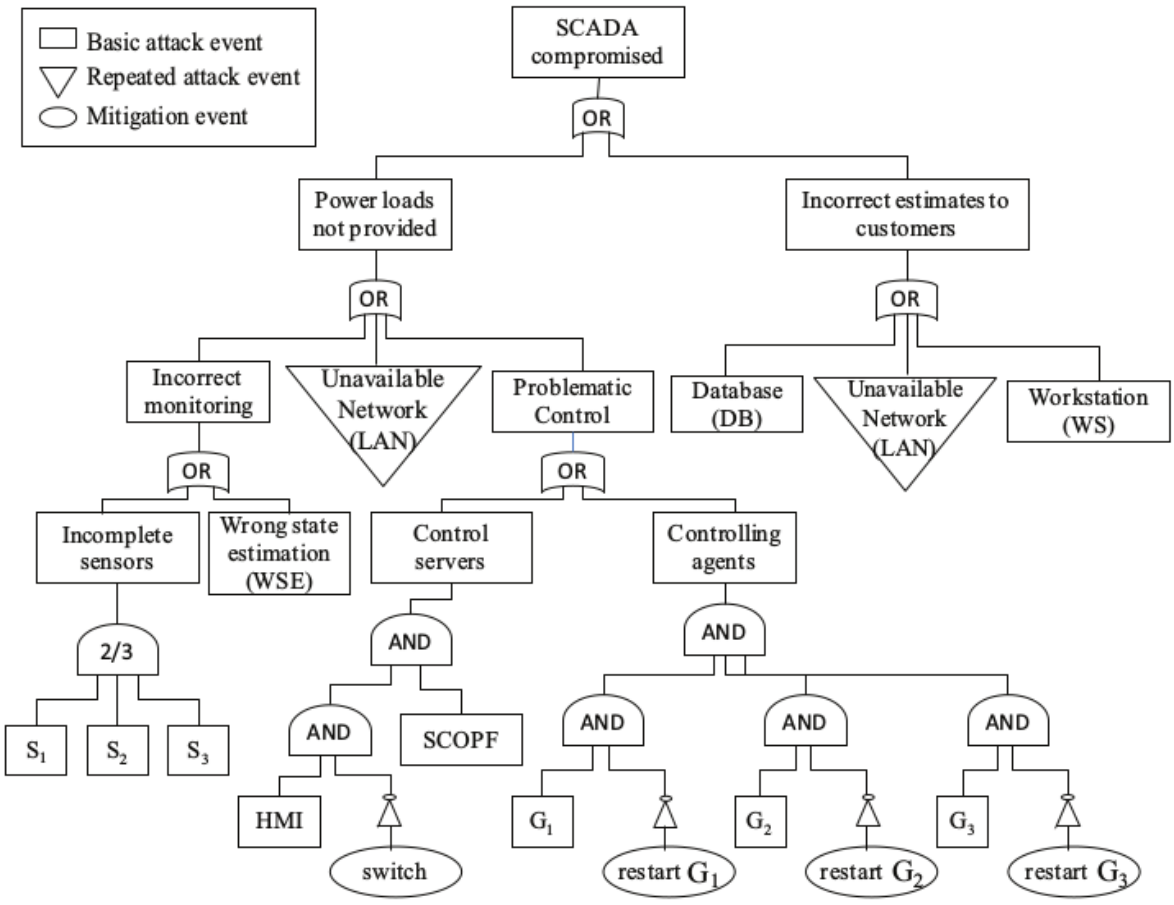

Figura 2: Grafo de ataques para un sistema SCADA [4]

- Requisitos que debe tener un atacante, tales como conocimientos, presupuesto económico, nivel de acceso a información interna.
- Si un ataque es legal o no.
- Comparar el efecto de diferentes contramedidas.
- Cómo emplear mejor un presupuesto para maximizar la mejora en materia de seguridad.

Todas esas cualidades atribuidas originalmente a los grafos de ataque, han demostrado

ser ciertas, y hay una gran variedad de modelos y metodologías de análisis de seguridad en la literatura que hacen uso de los grafos de ataque. No obstante, cabe preguntarse cómo es que, siendo todas esas cualidades ciertas, no están siendo utilizados de forma sistemática en organizaciones de todo tipo, habiendo pasado más de 15 años desde que se anunciaran por primera vez.

Lo cierto es que desde su concepción, los grafos de ataques se han tenido que enfrentar a diversos obstáculos de diferente tipo, que han hecho que aún a día de hoy no puedan ser aplicados de forma continua y satisfactoria en los departamentos de seguridad de las organizaciones, gestionar con el uso de estos la seguridad de las redes y equipos de los que se disponen.

El problema de la escalabilidad

Uno de los mayores problemas a los que se enfrentan los grafos de ataque, es a la escalabilidad de su uso en redes grandes. En un principio, los grafos de ataques se estaban generando y analizando de forma manual por expertos en seguridad. Sin embargo, hoy en día las redes de muchas organizaciones sobrepasan con facilidad el millar de dispositivos y generar grafos de ataques para tal número de objetivos potenciales de ataque, incluyendo para cada uno de ellos las diversas vulnerabilidades susceptibles de ser explotadas, y añadiendo los atributos adecuados que se utilizarán en el análisis, es absolutamente impensable para en términos del trabajo que eso supondría. Solamente es necesario un pequeño ejemplo para que podamos comprobar la magnitud: Si se supone una red con 1000 elementos diferentes, un promedio de 10 vulnerabilidades por elemento de la red, y un 20 nodos por cada vulnerabilidad, tendríamos 200.000 nodos en el grafo completo. Sin olvidar, además, que ésa sería solamente la primera parte de la tarea, puesto que una vez realizado ese paso, daría comienzo la segunda parte, consistente en analizar el grafo generado, estudiar las interacciones entre todos los elementos y obtener los resultados deseados según los criterios.

Este hecho no ha pasado desapercibido, y debido a ello gran cantidad de esfuerzos han sido dedicados a reducir dicho problema, buscando automatizar en la medida de lo posible las tareas asociadas a la utilización de esta técnica [5][6]. Originalmente se abordó la generación de grafos de ataque basándose en enumerar todos los posibles estados y generar los grafos de

ataque completos [7][8]. Estos modelos no son escalables en redes reales. Otros modelos han logrado generar grafos de ataques de redes con miles de elementos con buena escalabilidad [9], pero los grafos de ataques resultantes son demasiado grandes y complejos para que sean entendibles por una persona.

Aunque se hacen esfuerzos considerables en la literatura para abordar este problema, a día de hoy continúa siendo una cuestión que no se ha podido resolver de una manera satisfactoria, y por lo tanto sigue vigente.

2.1.1. Funcionamiento de los grafos de ataque y cuestiones a resolver

Para conocer y entender completamente el funcionamiento de los grafos de ataques y los retos actuales a los que se enfrenta, es necesario adquirir una visión global del flujo típico que se sigue cuando se trabaja con ellos. En la figura 3 se puede observar el proceso que se sigue durante el uso de los grafos de ataque. Este flujo consiste en las fases de Recolección de información, Construcción del grafo de ataques, Visualización y análisis.

Representación de los atributos y parámetros de los sistemas modelados

Otro de los obstáculos encontrados en la actualidad tiene su base en la problemática de la representación de los atributos y parámetros de los sistemas que deben ser modelados por los grafos de ataques. Un modelo de grafo de ataques debe ser capaz de representar toda la información necesaria sobre la red que está siendo objeto de estudio y las vulnerabilidades que pueden afectar a los elementos de la red, o a la misma red en su conjunto. Por otro lado, la inclusión de demasiadas características a tener en cuenta en el modelo, pueden incrementar la complejidad demasiado sin llegar a aportar información del todo relevante. Por lo tanto, encontrar el equilibrio adecuado que permita modelar todas las interacciones y propiedades de los ataques que van a ser objeto de análisis, no es una tarea fácil.

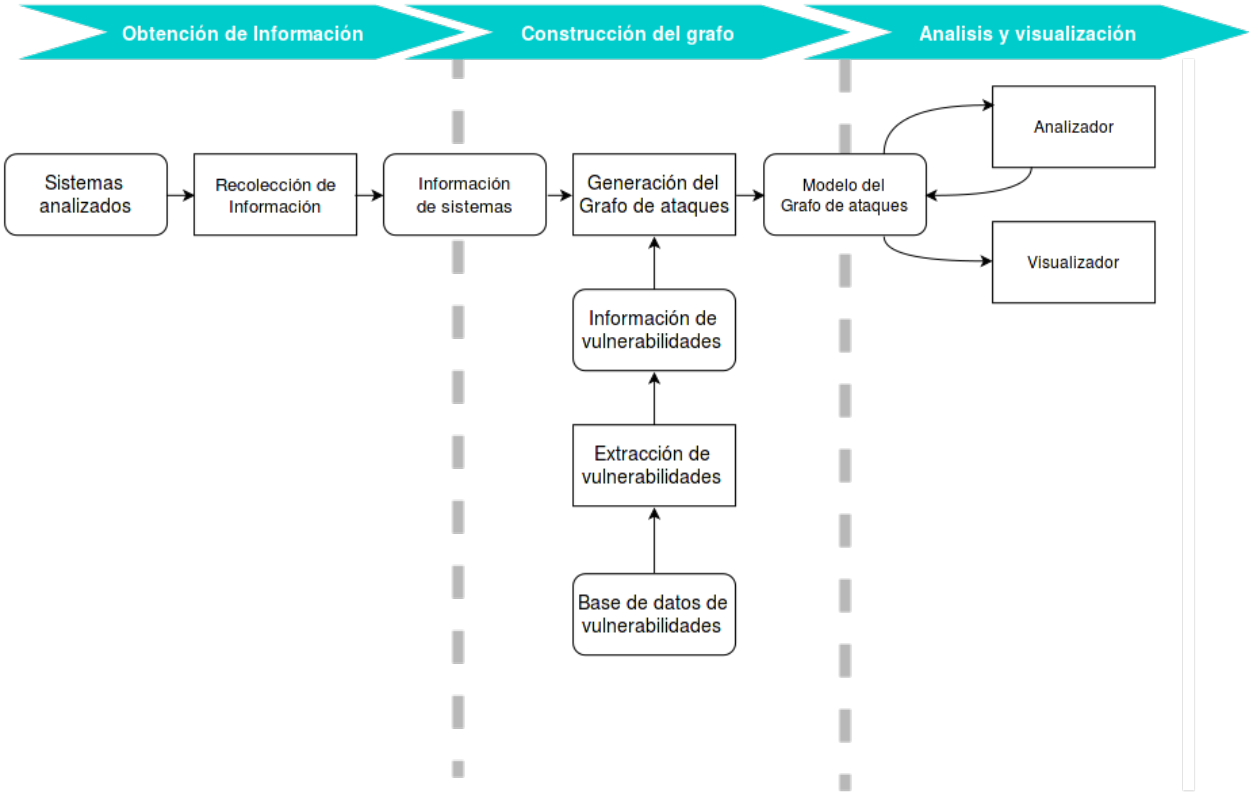

Figura 3: Flujo del proceso de generación y uso de grafos de ataques

Construcción de grafos de ataques

Obtener información precisa y de calidad acerca del sistema o red bajo riesgo de ataque, es un prerrequisito de la construcción de grafos de ataques. Dicha información tiene que incluir [10]:

- Topología de la red.
- Detalles de la configuración de cada elemento de la red.
- Tipos de servicios en ejecución en cada host.
- Políticas de seguridad.
- Relaciones de confianza.

- Descripciones de vulnerabilidades específicas que puedan ser explotadas por un atacante.

Esta información se puede adquirir desde un amplio rango de fuentes de información tales como escáneres de puertos, topología o vulnerabilidades; archivos de configuración de cortafuegos, ordenadores y routers; analizadores de políticas, bases de datos de vulnerabilidades e información proporcionada por usuarios y/o expertos. Esta información se adquiere tanto de forma automática como manual. Cuanto más detallada es la información recogida, más detallado es el grafo de ataques generado. Teniendo en cuenta la cantidad de fuentes de información que se tienen que utilizar para esta tarea, y la heterogeneidad de dichas fuentes, se puede deducir que el proceso de adquisición de la información necesaria para construir un grafo, dista de ser sencillo. Entre la dificultad de dicha tarea, son especialmente destacables por su complejidad, tanto el cálculo de qué elementos de la red son accesibles desde cuáles otros, como la obtención de descripciones de vulnerabilidades precisas.

La topología de la red determina qué otros hosts de una red son accesibles desde un host concreto. Esta información es muy útil durante la generación de grafos de ataque. Las herramientas que se utilizan en esta fase, pueden emplearla para relacionar diferentes elementos entre sí y deducir posibles caminos de un ataque. Dicha topología de la red tiene como posibles fuentes de datos a los informes de escáneres de red, reglas de cortafuegos, y descripciones de la estructura de la red mediante ficheros [11]. Para redes de organizaciones con un tamaño relativamente grandes, en las cuales existen miles de hosts y un buen número de firewalls, realizar el cálculo de la accesibilidad entre diferentes elementos de la red, resulta bastante costoso en tiempo de cómputo. Cada uno de dichos firewalls contiene de cientos a miles de reglas (reglas de control de acceso, reglas NAT...). Un escaneo sencillo llevado a cabo desde una dirección IP específica dentro de la red, ejercería el uso de solamente algunas de estas reglas, y es probable no lleguen a disparar aquellas que son aplicables a otras direcciones IP. En un caso real, es muy difícil saber qué direcciones IP origen y destino serían tratadas de forma diferente por el firewall. En redes de este tipo, utilizar escáneres de red supone una subestimación importante de la información de accesibilidad. Por otro lado, un simple escaneo de red tampoco es capaz de percibir el efecto de reglas de firewall que son dependientes

del tiempo para la accesibilidad. Es por ello que se convierte en algo imprescindible desccargar y analizar los archivos de configuración de los dispositivos de la infraestructura de red que realizan filtrado (firewalls, routers, switches, etc.) para poder determinar con precisión la accesibilidad entre elementos en subredes diferentes [2]. Existe, por lo tanto, una necesidad de desarrollar herramientas escalables para escanear la topología de red, que puedan llevar a cabo este tipo de análisis y proporcionar información de accesibilidad de la red adecuada a las herramientas de generación de grafos de ataque.

Se puede observar que, en la generación de grafos de ataque, las distintas piezas de información necesarias provienen desde fuentes de diferentes y se presentan en formas semánticamente diferentes las unas de las otras. Aunque las diferentes unidades de información representan aspectos diferentes de la seguridad de los sistemas objeto de análisis, toda esa información necesita ser unida de la forma apropiada. Para ello, es imprescindible utilizar las estructuras de datos adecuadas. La información recabada puede ser estática o dinámica. Por ejemplo, pueden producirse cambios en la topología de la red cuando un dispositivo se conecta a, o se desconecta de ella; o pueden instalarse, desinstalarse, o incluso reconfigurarse nuevos componentes de software en los sistemas. Por otro lado, las descripciones de vulnerabilidades suelen ser reconocidas como información de tipo estático, ya que una vulnerabilidad anunciada, no deja de tener vigencia. Se hace necesario, por lo tanto, disponer de una estructura de datos unificada mediante la que tratar dicha información. También es necesario considerar si la información se presenta de forma fuertemente o débilmente acoplada entre sí. Es necesario encontrar las estructuras de datos adecuadas que permitan reunir toda esta información en la generación de grafos de ataques [10].

Cuando ya se ha reunido toda la información requerida como entrada para la construcción de un grafo de ataques, y ha sido unificada adecuadamente, se pasa a la fase de construcción del grafo de ataques. Esto se realiza habitualmente siguiendo estos pasos: Se computa una matriz de accesibilidad o alcanzabilidad, utilizada para determinar el conjunto de hosts alcanzables desde otro cualquiera. Junto a esto, la información sobre las vulnerabilidades de cada elemento se integran en la matriz anterior, dando como resultado un grafo de ataque. En *A Taxonomy for Comparing Attack-Graph Approaches*, Heberlein et Al.[12] presentan

una serie de características para comparar diferentes modelos de grafos de ataques según estas:

- Ataques monotónicos o no monotónicos: Se consideran ataques monotónicos aquellos en los que el orden en el que se lleva a cabo un ataque concreto no puede impedir realizar otro que previamente era posible realizar. Esto significa que una vez que dos ataques pueden llevarse a cabo, el orden en el que se hagan no importa. Por el contrario, los ataques no monotónicos pueden dar esa situación, y realizar un ataque puede suponer que otro ataque previamente realizable no lo sea. Desde el punto de vista de la construcción de un grafo de ataque, un modelo con ataques no monotónicos implica una mayor cantidad de estados que deberán generarse y explorarse.

- Camino único o Todos los caminos: Un modelo de camino único simplemente mostrará uno de los caminos en el grafo que permiten a un atacante alcanzar su objetivo. Por otro lado, uno que genere todos los posibles caminos, es más costoso de computar, pero permiten hacer un análisis para identificar las vulnerabilidades con mayor incidencia, el conjunto mínimo de contramedidas que podría proteger la red, etcétera.

- Encadenamiento hacia adelante o hacia atrás: Un modelo con encadenamiento hacia adelante empezaría por los nodos hojas desde donde puede comenzar un ataque, y explorar dicho grafo hasta encontrar un objetivo alcanzable desde dicho nodo. El encadenamiento hacia atrás, por el contrario, comienza en un objetivo, y analiza precondiciones para encontrar precondiciones que permitirían dicho ataque.

Estas características pueden utilizarse para comparar o categorizar diferentes modelos de grafos de ataques.

Para la realización de un ataque, el atacante debe cumplir ciertas precondiciones sobre el estado de la red, y tras realizarlo con éxito, nuevas condiciones conocidas como postcondiciones son añadidas al estado de la red. En este contexto, la definición de precondiciones se corresponde con «*los privilegios que el atacante tiene a su disposición sobre equipos de la red antes de lanzar un ataque*», mientras que la definición de postcondiciones se corresponde con «*los privilegios que el atacante posee sobre los equipos de la red tras haber llevado a cabo un ataque con éxito*». En los grafos de ataques, una postcondición de un ataque concreto,

actúa como precondición para el ataque siguiente, permitiendo que se suceda una secuencia de ataques. Si no se encadenan de manera adecuada en el grafo de ataque las precondiciones y las postcondiciones de los diferentes nodos, el grafo de ataques obtendrá resultados incorrectos. Por lo tanto, otra cuestión muy importante durante la construcción de grafos de ataque es establecer de forma adecuada esa cadena de privilegios de las precondiciones y las postcondiciones para que los resultados sean adecuados.

Visualización de los grafos de ataques

Cuando un grafo de ataques es generado, se pone a disposición de un analista de seguridad que se encargará de interpretarlo y entender posibles intrusiones en la red y los pasos que habilitarían dicha intrusión. Aún cuando los grafos de ataques están concebidos para su uso por analistas de seguridad de forma que se puedan realizar estudios sobre la seguridad de la red, los enfoques actuales de cara a la visualización de estos presenta algunas carencias en la práctica. Algunas de ellas son:

- Al ser las redes de comunicaciones de organizaciones unas infraestructuras altamente interconectadas y complejas, los grafos de ataques resultantes presentan una gran sobrecarga de información díficil de interpretar.

- Habitualmente, un único ataque involucra a un equipo inicial, un equipo final, y uno o más equipos intermedios. Por lo tanto, esto hace que el número de nodos y aristas del grafo crezcan de forma combinatoria (como mínimo cuadrática en el número de hosts, multiplicado por el número de vulnerabilidades) [10]. Esto implica que para una red de una organización con un tamaño razonablemente grande, el grafo de ataques es de un tamaño descomunal, y resulta incomprensible para un ojo humano. Un análisis manual de un grafo tan grande y complejo, es una empresa casi imposible, y lo único que se consigue es complicar el proceso de obtener información útil. Para una red lo suficientemente grando como para disponer de más de mil equipos, la explosión de tamaño del grafo se convierte en algo inevitable. En la figura 4 se puede observar una ilustración gráfica de este hecho. El grafo de ataques representado en dicha figura se

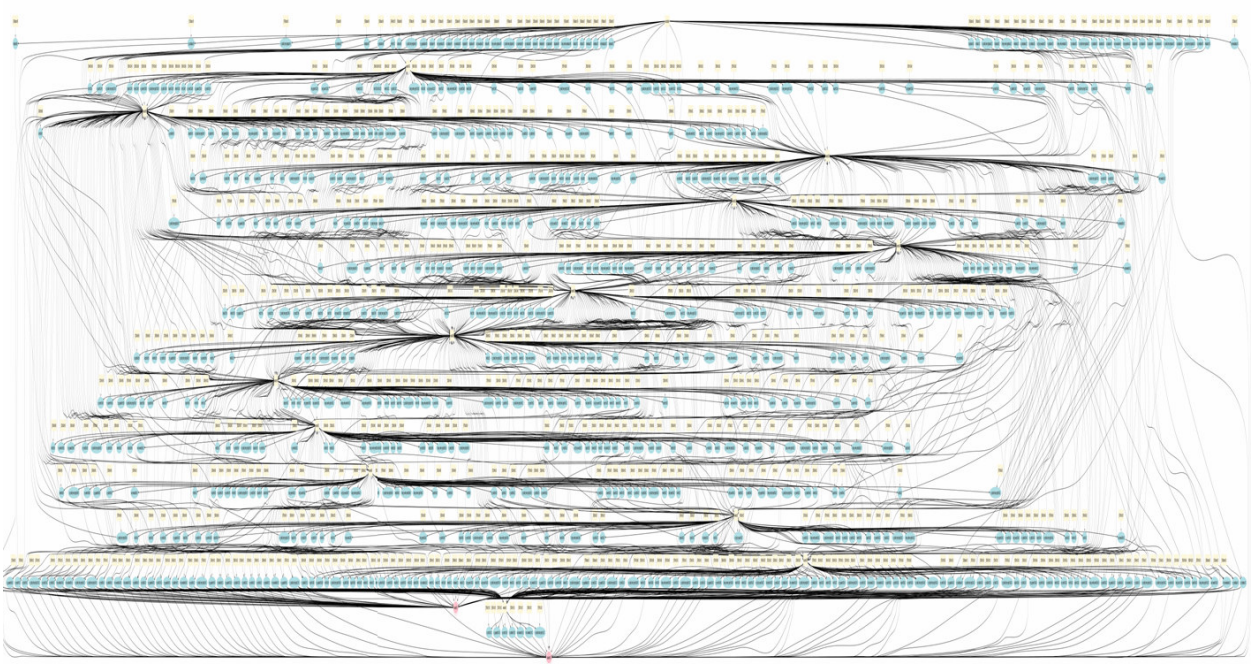

Figura 4: Grafo de ataques completo para una red de tan solo 17 hosts [13].

corresponde con una red de tan solo 17 equipos, y ya resulta demasiado grande para ser comprensible.

- Otro de los problemas de los grafos de ataque, es que no representan de una forma adecuada la topología y accesibilidad de la red subyacente. Esto implica que es realmente complicado establecer una relación entre los caminos de los posibles ataques y la evolución de dicho ataque en la topología de la red. No resulta sencillo saber de una forma rápida cuáles son los equipos que permiten que cierto ataque pase de una subred a otra, o por qué un camino concreto de ataque es factible.

- En muchos de los modelos estudiados, se generan grafos de ataques completos que muestran todos los posibles escenarios de ataque, tanto los que culminan en un ataque con éxito al objetivo vulnerable, como aquellos caminos y estados que llevan a puntos muertos por parte del atacante, y no consiguen su objetivo. Estos grafos de ataques completos poseen nodos redundantes, y como se ha visto antes son demasiado complejos de gestionar y comprender para el administrador. Aquellos a cargo de llevar a

cabo el estudio de los grafos de ataque, solamente están interesados en conocer posibles escenarios de ataque con éxito, y por lo tanto, disponer de nodos que no son útiles para conocer amenazas a la red carece de sentido. Por otro lado, en una red real, una gran cantidad de equipos contienen características y configuraciones muy similares (cuando no idénticas). Esto también introduce una cantidad importante de redundancia. Se puede decir, por tanto, que los grafos de ataques arrastran una cantidad amplia de información duplicada, que no solo impide una correcta representación de estos, si no que también supone un impedimento a la hora de la computación durante la construcción y el análisis.

La capacidad de producir una forma efectiva de visualización de los grafos de ataques, es una cuestión esencial para su utilidad. En la literatura existen múltiples maneras de representar los grafos de ataques de forma visual. La mayoría de ellos se producen por la necesidad de poder generar grafos de ataques de manera escalable, y otros pretenden facilitar el análisis de dichos grafos[14]. Se han utilizado diferentes técnicas para hacerlos más comprensibles, tales como Agrupamiento, Agregación, Clustering, Mapas de árbol, Compresión de grafos y Priorización.

Los enfoques anteriores se basan en la generación inicial del grafo, y después aplicar técnicas que permitan reducir su complejidad visual. Incluso con estas mejoras en la complejidad visual de los grafos, la dificultad de relacionar grafos de ataque con las topologías subyacentes sigue estando vigente. Las técnicas empleadas hasta ahora no permiten establecer relaciones entre los elementos observables en un grafo de ataques, y aquellos presentes en la red (firewalls, jerarquías de subredes), ya que éstos últimos no se encuentran representados en dicho grafo. Quedan aún, por lo tanto, algunas cuestiones por resolver, que los intentos anteriores de producir mejoras en la visualización de los grafos de ataques no han tenido en cuenta:

- Un grafo de ataques comprimido debe presentar información lo más útil posible para los analistas de seguridad encargados de analizarlo, y presentar la información sobre las amenazas a los sistemas tan bien como el original.

- Un grafo de ataques comprimido no debe conllevar pérdida de información. Especialmente aquella referida a las secuencias de ataque que puede sufrir la red.

Un grafo de ataques capaz de mostrar la información de este modo, mejoraría la capacidad de un analista de seguridad de aprovechar la información disponible en el grafo de ataques y usar dicha información para incrementar los niveles de seguridad de la red.

Análisis de los grafos de ataques

Una vez que se han generado los grafos de ataques, los administradores de la seguridad de la red deben encargarse de analizar dicho grafo, y obtener resultados provechosos, como podrían ser el número de posibles caminos de ataque, el camino de ataque más corto, el más probable de suceder. Los resultados de dicho análisis se utilizan para tomar las medidas adecuadas en materia de protección de la red.

Para asistir a un administrador en su tarea de abordar posibles problemas de seguridad en una red, un grafo de ataques puede generar recomendaciones de seguridad. Este tipo de recomendaciones pueden ser: sugerenncias de cambios en la topología de la red, parches para vulnerabilidades en el software instalado, bloqueo de aquellos puertos abiertos innecesarios, etcétera. Ésta es un área de investigación muy prominente puesto que abre un abanico de posibilidades al facilitar al administrador su tarea enormemente.

Almacenamiento de los grafos de ataques

Una vez que un grafo de ataques ha sido generado, es necesario almacenarlo para su uso posterior durante las fases de análisis. Un grafo de ataques de unar red de miles de equipos, puede alcanzar un tamaño muy grande. Este grafo de ataques tendrá que ser almacenado y recuperado de una forma eficiente, para que los procesos de análisis que requieran su uso puedan realizarlo fácilmente y sobre la marcha. La mayor parte de los enfoques estudiados para este trabajo, no hacían mención alguna a la cuestión del almacenamiento y recuperación de los datos de grafos de ataque.

2.2. Vulnerabilidades

En esta sección se va a mostrar qué son las vulnerabilidades, cómo se utilizan en los grafos de ataques, así como bases de datos de vulnerabilidades de las más importantes. Los grafos de ataques se basan en la información sobre vulnerabilidades de sistemas y protocolos para poder elaborar los caminos que un atacante utilizaría para comprometer la seguridad de los sistemas. Las bases de datos proporcionan información sobre vulnerabilidades existentes, nuevas que surgen, etcétera.

Una vulnerabilidad se define como un defecto de software que permite a un atacante violar una política de seguridad explícita o implícita y lograr un impacto o consecuencia [15]. Disponer de información precisa y adecuada sobre vulnerabilidades es fundamental para la generación de grafos de ataques, puesto que conocer cómo un atacante lleva a cabo la explotación de dichas vulnerabilidades, es lo que permitirá establecer medidas de seguridad adecuadas para prevenir, detectar y mitigar dichos ataques.

Para lograr una modelación adecuada de los caminos de ataque en un grafo, es imprescindible disponer de descripciones adecuadas de vulnerabilidades, así como las precondiciones y postcondiciones derivadas de ellas. Estas descripciones incluyen información que permita conocer los recursos disponibles para un atacante antes de lanzar un ataque, y los que obtiene después de haberlo lanzado; y cómo se encadenarían unas vulnerabilidades con otras formando ataques más complejos en cadena.

2.2.1. Bases de datos de vulnerabilidades

Las bases de datos de vulnerabilidades comprenden grandes recopilaciones de debilidades de software, y se encargan de mantener y administrar dichas recopilaciones lo más actualizadas posible, y de forma que sean útiles. Éstas bases de datos se presentan de dos formas diferentes: de acceso público y comerciales. Una de las bases de datos de vulnerabilidades públicas más importante y reconocida es la National Vulnerability Database (NVD) [16].

National Vulnerability Database

La NVD es una base de datos perteneciente al gobierno de Estados Unidos. Esta base de datos registra vulnerabilidades desde 1988. Actualmente se basa en la notación proporcionada por la CVE (Common vulnerabilities and exposures) para la identificación de las vulnerabilidades. A día de hoy esta basee de datos incluye información sobre más de 70.000 vulnerabilidades, incorporando una media de 580 cada mes. Toda la información contenida en esta base de datos se pone a disposición de todo aquel que desee utilizarla sin ningún tipo de condiciones, porque se distribuye como dominio público. Esto permite que se utilice para cualquier tipo de fin, ya sea académico, comercial o de otro tipo.

Esta base de datos almacena su información utilizando formatos específicamente diseñados para que la extracción y análisis de los datos en ella contenidos, puedan realizarse mediante el uso de herramientas automatizadas. Entre dichos formatos encontramos con CVSS(Common Vulnerability Scoring System). Se trata de un estándar abierto para clasificar la gravedad de una vulnerabilidad de software entre 0 y 10, siendo la gravedad más alta conforme aumenta la puntuación. Proporciona además puntuaciones adicionales que permiten conocer si una vulnerabilidad compromete confidencialidad, integridad y disponibilidad; si se requiere autenticación para explotarla, etc.

Otro formato destacable de NVD es la base de datos XML de las vulnerabilidades. La base de datos de vulnerabilidades se puede descargar desde NVD. Ésta se proporciona en formato XML, siguiendo un esquema que se puede descargar de la propia web de la base de datos.

En el listado de código siguiente se puede ver una vulnerabilidad a modo de ejemplo de la base de datos. Dicha vulnerabilidad se corresponde con un problema de los routers cisco, en ciertas versiones del sistema cisco sobre ciertas versiones de hardware.

- En primer lugar se observan las propiedades que describen qué equipos son susceptibles de ser atacados por esta vulnerabilidad. Son aquellos que poseen cualquiera de

las propiedades del primer grupo, unida a cualquiera de las propiedades presentes en el segundo grupo. Dichas propiedades, son identificadores que sirven para representar versiones específicas del software y hardware en cuestión. Además se utilizan unos operadores booleanos que no dejan ambigüedad posible en la definición de a qué componentes afecta.

- Se incluye una entrada con el identificador CVE de la vulnerabilidad que está siendo descrita.
- Después se puede observar la puntuación de la vulnerabilidad en las diferentes métricas que proporciona el formato CVSS. Se observa que tiene una puntuación global de 7'8, lo que le otorga una gravedad alta a la vulnerabilidad. Se definen también los vectores de acceso, y la complejidad de efectuar el acceso; se indica si la vulnerabbilidad requiere autenticación, y por último el nivel de impacto sobre la triada CIA de cualidades de seguridad.
- Se incluyen también referencias en otros medio a la vulnerabilidad en cuestión. En este caso la referencia es al propio fabricante de los sistemas afectados.
- Por último, encontramos un resumen en texto del ataque, a qué sistemas afecta, cómo se produce y cuáles son sus consecuencias.

Listado 1: Ejemplo de vulnerabilidad de NVD

```
<entry id="CVE-2015-6272">
  <vuln:vulnerable-configuration id="http://www.nist.gov/">
    <cpe-lang:logical-test operator="AND" negate="false">
      <cpe-lang:logical-test operator="OR" negate="false">
        <cpe-lang:fact-ref name="cpe:/o:cisco:ios_xe:2.1.0"/>
        <cpe-lang:fact-ref name="cpe:/o:cisco:ios_xe:2.1.1"/>
        <cpe-lang:fact-ref name="cpe:/o:cisco:ios_xe:2.1.2"/>
        <cpe-lang:fact-ref name="cpe:/o:cisco:ios_xe:2.1.3"/>
        <cpe-lang:fact-ref name="cpe:/o:cisco:ios_xe:2.2.1"/>
        <cpe-lang:fact-ref name="cpe:/o:cisco:ios_xe:2.2.2"/>
```

```
            <cpe-lang:fact-ref name="cpe:/o:cisco:ios_xe:2.2.3"/>
            <cpe-lang:fact-ref name="cpe:/o:cisco:ios_xe:2.3.0"/>
          </cpe-lang:logical-test>
          <cpe-lang:logical-test operator="OR" negate="false">
            <cpe-lang:fact-ref name="cpe:/h:cisco:asr_1001:-"/>
            <cpe-lang:fact-ref name="cpe:/h:cisco:asr_1001-x:-"/>
            <cpe-lang:fact-ref name="cpe:/h:cisco:asr_1002:-"/>
            <cpe-lang:fact-ref name="cpe:/h:cisco:asr_1002-x:-"/>
            <cpe-lang:fact-ref name="cpe:/h:cisco:asr_1004:-"/>
            <cpe-lang:fact-ref name="cpe:/h:cisco:asr_1006:-"/>
            <cpe-lang:fact-ref name="cpe:/h:cisco:asr_1013:-"/>
          </cpe-lang:logical-test>
        </cpe-lang:logical-test>
      </vuln:vulnerable-configuration>
      <vuln:vulnerable-software-list>
        <vuln:product>cpe:/o:cisco:ios_xe:2.1.0</vuln:product>
        <vuln:product>cpe:/o:cisco:ios_xe:2.2.3</vuln:product>
        <vuln:product>cpe:/o:cisco:ios_xe:2.1.1</vuln:product>
        <vuln:product>cpe:/o:cisco:ios_xe:2.1.2</vuln:product>
        <vuln:product>cpe:/o:cisco:ios_xe:2.2.1</vuln:product>
        <vuln:product>cpe:/o:cisco:ios_xe:2.3.0</vuln:product>
        <vuln:product>cpe:/o:cisco:ios_xe:2.1.3</vuln:product>
        <vuln:product>cpe:/o:cisco:ios_xe:2.2.2</vuln:product>
      </vuln:vulnerable-software-list>
      <vuln:cve-id>CVE-2015-6272</vuln:cve-id>
      <vuln:published-datetime>2015-08-31T16:59:06.280-04:00</
         vuln:published-datetime>
      <vuln:last-modified-datetime>2015-09-01T14:26:55.103-04:00</
         vuln:last-modified-datetime>
      <vuln:cvss>
        <cvss:base_metrics>
          <cvss:score>7.8</cvss:score>
          <cvss:access-vector>NETWORK</cvss:access-vector>
          <cvss:access-complexity>LOW</cvss:access-complexity>
          <cvss:authentication>NONE</cvss:authentication>
          <cvss:confidentiality-impact>NONE</cvss:confidentiality-impact>
          <cvss:integrity-impact>NONE</cvss:integrity-impact>
          <cvss:availability-impact>COMPLETE</cvss:availability-impact>
```

```
        <cvss:source>http://nvd.nist.gov</cvss:source>
        <cvss:generated-on-datetime>2015-09-01T07:40:49.847-04:00</
           cvss:generated-on-datetime>
      </cvss:base_metrics>
    </vuln:cvss>
    <vuln:cwe id="CWE-399"/>
    <vuln:references xml:lang="en" reference_type="VENDOR_ADVISORY">
      <vuln:source>CISCO</vuln:source>
      <vuln:reference href="http://tools.cisco.com/security/center/
         viewAlert.x?alertId=40689" xml:lang="en">20150827 Cisco ASR
         1000 Series Aggregation Services Routers Crafted H.323 Packet
         Denial of Service Vulnerabilities</vuln:reference>
    </vuln:references>
    <vuln:summary>Cisco IOS XE 2.1.0 through 2.2.3 and 2.3.0 on ASR
       1000 devices, when NAT Application Layer Gateway is used, allows
        remote attackers to cause a denial of service (Embedded
       Services Processor crash) via a crafted H.323 packet, aka Bug ID
        CSCsx35393, CSCsx07094, and CSCsw93064.</vuln:summary>
  </entry>
```

Cabe destacar que dichas entradas de la base de datos con un formato tan específico, suponen un esfuerzo destacable hacia la automatización de la recopilación de este tipo de información. El hecho de que la NVD provea su información con un formato XML tan bien definido, simplifica bastante el esfuerzo necesario para hacer un analizador sintáctico que pueda extraer de forma autónoma dicha información.

Sin embargo, no toda la información puede ser extraída de forma automática en base a los datos y el formato obtenido de las bases de datos. En el ejemplo de la vulnerabilidad anterior, se ha visto que mediante el análisis de los datos automatizados, se podría obtener con relativa precisión la gravedad de la vulnerabilidad, en base a la métrica CVSS, así como algunas de las precondiciones necesarias para explotarla (versiones específicas de hardware y software afectados). Por desgracia, para conocer con más detalle qué otras características son necesarias para el ataque, así como las postcondiciones que proporcionaría, es imprescindible entender el resumen adicional proporcionado, y en muchos casos, incluso buscar referencias

adicionales.

Algunas soluciones para la extracción y unificación de información de vulnerabilidades de forma automática han sido presentadas, y llegan a analizar los comentarios de texto y extraer cierta información útil de ellos [17]. Pese a todo, sigue siendo necesaria la intervención manual para el correcto análisis de vulnerabilidades y generación de sus propiedades, lo que dificulta su generación de forma automática.

Tipos de vulnerabilidades

Ser capaces de organizar las diferentes vulnerabilidades que aparecen en referencia a una serie de cualidades comunes y dividirlas en tipos, es una cuestión imprescindible. Al realizar esto, se permite identificar puntos comunes entre vulnerabilidades, que suelen ser útiles tanto de cara a conocer el problema que las originó en primer lugar, como a la hora de encontrar soluciones conociendo cómo se solucionan otras del mismo tipo. La base de datos de meta-vulnerabilidades CVE proporciona una categorización de las vulnerabilidades en 37 tipos diferentes. Las más frecuentemente encontradas siendo: denegación de servicio (dos), desbordamiento de pila (overflow), cross site scripting (xss). Los diferentes modelos analizados en la literatura, utilizan en ocasiones sus propias categorías diferentes de las proporcionadas en CVE.

La definición y categorización de las vulnerabilidades en tipos con estructuras jerárquicas puede resultar muy beneficiosa a la hora de tratar con dichas vulnerabilidades. Una estructura jerárquica permitiría clasificar las vulnerabilidades en tipos amplios, y que luego dichos tipos incluyesen a su vez subtipos que tienen al supertipo como denominador común. Dicha taxonomía jerárquica puede facilitar la tarea de tratar de forma automatizada con las vulnerabilidades, al agrupar las vulnerabilidades en un pequeño número de grupos grandes, a los cuales se les puede realizar un tratamiento inicial en masa, y luego recibir un tratamiento más específico según el subtipo de vulnerabilidad.

Sin embargo, esto no es fácil de realizar, ya que sería necesario poder establecer una distinción clara y unas categorías que no dieran lugar a posibles ambigüedades. Esta tarea dista

de ser algo sencillo y trivial, puesto que incluso la categorización totalmente disjunta que lleva a cabo CVE tiene en ocasiones una diferenciación difusa entre algunas de sus categorías.

2.3. Eventos

En este apartado se explora el concepto de evento, y cómo podemos utilizarlos en nuestro sistema. Los eventos van a permitirnos representar cambios en las redes que van a ser objeto de estudio, así como cambios producidos por los propios modelos de grafos de ataque durante su funcionamiento.

Un evento se define como algo que sucede en un momento determinado y que tiene un carácter poco común, o de algún modo excepcional. Para que sea algo fuera de lo común poseerá una serie de cualidades que le confieren dicho estatus. Estas cualidades representan información que resulta útil. En el campo de la informática, los eventos son utilizados de forma habitual para modelar las acciones de un usuario detectadas por un componente de software, y que deben recibir algún tipo de tratamiento; el paso de mensajes entre diferentes subsistemas; o la detección de sucesos anómalos que deben ser analizados.

En el ámbito que se está tratando en este trabajo, un evento puede comprender desde un cambio en la topología de la red (un nuevo equipo conectado, o incluso cambios de configuración), a nuevas vulnerabilidades descubiertas, e incluso ataques. Por ese motivo, se han definido una serie de categorías de eventos de alto nivel:

- Vulnerabilidades: Estos eventos se corresponderán con información recibida sobre vulnerabilidades.
- Topología de red: Cambios producidos en la topología de red, pueden contemplar desde la conexión y desconexión de elementos, hasta instalación de elementos software o cambios de configuración.
- Soluciones: Soluciones aportadas por los modelos de grafos de ataques tras realizar sus análisis.

Dichas categorías son lo suficientemente específicas para ser descriptivas de por sí mismas, lo que permite un tratamiento rápido de eventos según su categoría. Al mismo tiempo, son lo suficientemente abstractas y de alto nivel, como para evitar la ambigüedad y estar bien acotado qué eventos pertenecen a ellas y cuáles no. Tanto esa categorización como las subcategorías, podrían extenderse con facilidad en el futuro si fuera necesario.

3. Diseño y resolución del trabajo realizado

En esta sección se estudiará el diseño del simulador propuesto y desarrollado en este trabajo. Dicho simulador pretende proporcionar una manera de estudiar el rendimiento y las diferencias entre diferentes enfoques propuestos en la literatura para trabajar con los grafos de ataques en redes, probándolos y analizándolos de una manera similar sobre el mismo escenario de red, y comprobando los resultados que ofrecen.

Se estudiarán aquellos conceptos necesarios para el entendimiento de la arquitectura propuesta, tales como el flujo de datos que recorre la aplicación, desde que un suceso es detectado y registrado por la aplicación hasta los cambios que produce sobre los módulos de toma de decisiones, y cómo eso se traduce en cambios sobre la visualización de los grafos y la información que estos proporcionan. Se verán también los detalles del diseño y la implementación de la aplicación, y cómo se han establecido los diseños en la práctica, y qué problemas se han encontrado(ver 3.1). Se estudiarán también los modelos de grafos de ataques incluidos en esta primera versión del simulador(ver 3.2). Por último, se analizarán los resultados obtenidos(ver 3.3).

3.1. Arquitectura e implementación del simulador

En esta subsección serán explicadas las diferentes partes que componen nuestro sistema, incluyendo aquellos componentes que forman la visión de la arquitectura, y las definiciones necesarias para la implementación de modelos que puedan utilizarse en nuestro simulador.

El desarrollo realizado a lo largo de este proyecto, presenta una serie de módulos bien definidos, que podemos observar en la figura 5. Estos módulos son independientes entre sí, y se comunican mediante interfaces bien definidas, lo que permite que puedan ser desarrollados de forma individual sin que sus cambios internos afecten al resto de la arquitectura, siempre que se mantenga el uso de dichas interfaces.

A modo de resumen general de la estructura del simulador presentado en este trabajo, podemos observar un módulo que es el flujo de eventos. Este módulo será el encargado de

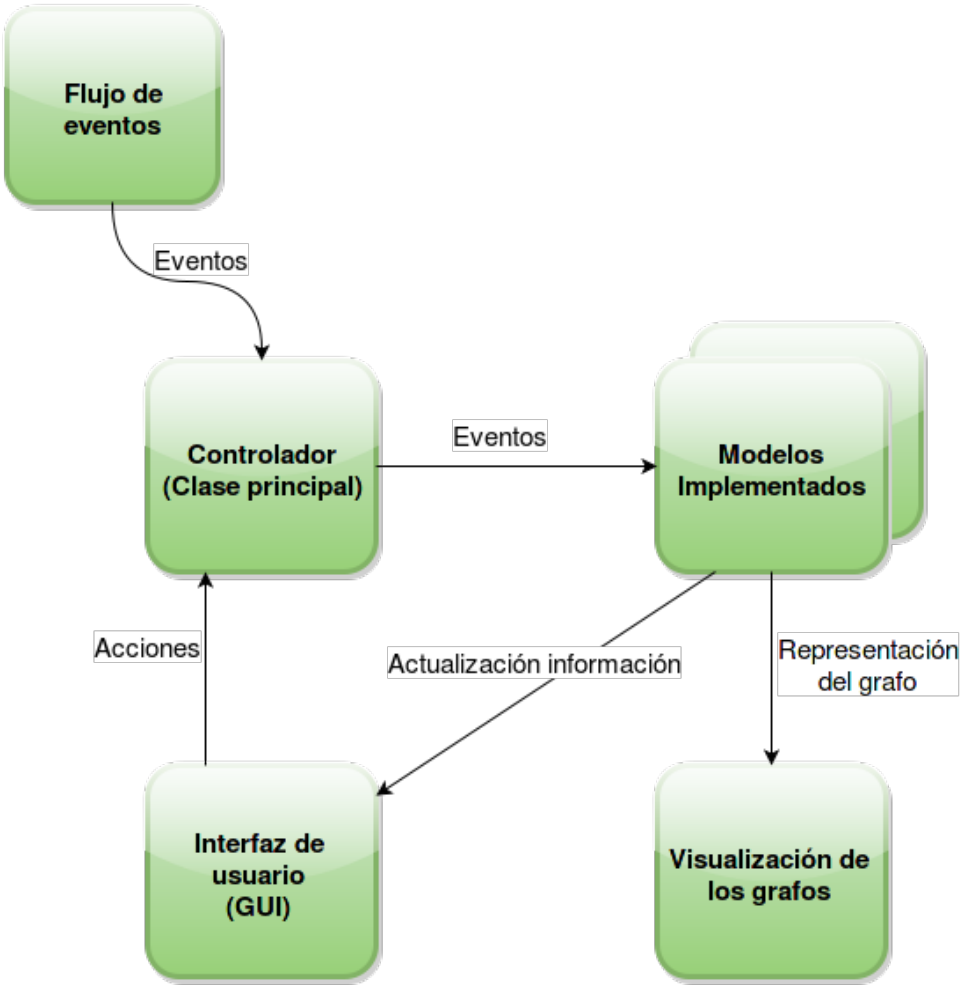

Figura 5: Visión global de la arquitectura del simulador

recibir información sobre sucesos que estén teniendo lugar en la red, y procesarlos para generar los eventos adecuados correspondientes al suceso que está teniendo lugar. Después, dichos eventos son transferidos al controlador, quien los deberá interpretar y llevar a cabo la acción adecuada. Estos eventos pueden contener información sobre diversos sucesos tales

como modificaciones en la topología de red, vulnerabilidades, etc.

En el siguiente paso encontramos al controlador. Este controlador es el encargado de de establecer un punto intermedio entre los diferentes módulos, y su función es principalmente la de ser un coordinador de las comunicaciones entre los demás módulos. Esta clase principal recibirá los eventos proporcionados por los flujos de eventos, los analizará e identificará a qué módulo o módulos debe remitirle dicha información, para finalmente efectuar dicha transmisión de información. Este módulo permanecerá en el centro de buena parte de las interacciones que tengan lugar en el simulador. Debido a su posición central como distribuidor de la comunicación, este módulo es el lugar ideal para ejercer el rol de la inicialización y parada del resto de módulos. Será el encargado de arrancar todos los elementos necesarios durante la fase de inicialización, y también de pararlos al finalizar, o cuando sea necesario.

En la figura 5 se puede observar que bajo el controlador, encontramos la interfaz de usuario. Ésta se encargará de mostrar al usuario en pantalla información variada sobre los sucesos que tienen lugar en el simulador, y el funcionamiento de los modelos. La interfaz de usuario enviará información de las acciones efectuadas por el usuario al controlador, quien las procesará de forma adecuada. También permitirá al usuario elegir, de entre los diferentes modelos de grafos de ataque implementados, cuál utilizar en cada momento.

A la derecha del controlador se pueden observar los diferentes módulos que se corresponden con los modelos de los grafos de ataque implementados. Estos módulos deberán comunicarse con el controlador, recibir eventos de los tipos adecuados de éste último, y realizar sus cálculos y computación para analizar los grafos de ataques y producir los resultados adecuados. Estos módulos son, sin lugar a dudas, una de las partes más complejas del simulador. Para que la creación de nuevos módulos sea lo más sencilla posible, y al mismo tiempo permitir al simulador conocer los detalles del modelo que sean necesarios, se han diseñado una interfaces de comunicación y de configuración que permitan a ambos interactuar, sin necesidad de conocer a priori todas las necesidades de un módulo.

Por último se puede ver el módulo de visualización y representación de los grafos de ataques. Este módulo se encarga de ofrecer al usuario una representación gráfica del grafo de ataque que realiza un modelo. Está íntimamente ligado a los módulos de modelos de grafos de ataques. Aunque se trata de una interfaz gráfica, se ha diseñado independientemente de la interfaz gráfica principal del simulador para que se la pueda ligar de forma adecuada a las

necesidades de los modelos, sin que deba ligarse el funcionamiento del propio simulador. Este módulo presenta una interfaz sencilla para crear una representación de grafos de ataques que los modelos pueden crear durante la ejecución, para la visualización por parte del usuario.

Si se presta atención a la estructuración de las comunicaciones entre las diferentes partes del sistema, puede ver que se ha seguido la estructura del diseño conocida como Modelo-Vista-Controlador, que permite separar de forma adecuada los módulos, y reducir el acoplamiento.

En cuanto a lo concerniente a los conceptos básicos sobre la implementación de esta arquitectura, es notable destacar que el lenguaje de programación elegido para este desarrollo ha sido Java, en su versión 1.8. Esto ha facilitado que la implementación se lleve a cabo y se pruebe en plataformas diferentes (Microsoft Windows y Arch Linux).

3.1.1. Módulo de flujo de eventos

Este módulo supone en cierta medida la puerta de entrada al simulador de información externa sobre diferentes elementos, aunque no será la única. Recibirá información sobre los eventos que tengan lugar, y los enviará a la clase principal de control. Por lo tanto, se encargará de generar o eventos sobre información recibida, y transmitirlos al núcleo del simulador.

El simulador ha sido diseñado con el objetivo de poder recibir información heterogénea desde diferentes fuentes, que permitan reflejar cambios en la topología de red del escenario, en las vulnerabilidades conocidas, e incluso cambios sobre las contramedidas aplicadas. Dicha información podría proceder de sistemas de detección de intrusiones en red (NIDS), de sensores, de bases de datos de vulnerabilidades, etc. Como se vio durante el análisis del estado del arte en las secciones 2.1.1 y 2.2.1, recibir información tan heterogénea y hacer un análisis y tratamiento correcto y útil de forma automatizada, no es una cuestión baladí.

El diseño del flujo de eventos permitiría la recepción de información desde múltiples módulos, pudiendo implementarse diferentes módulos específicos según las fuentes de información. Otros flujos de información más sencillos y acotados, aunque más limitados, también son posibles, y son los que se han implementado en la primera versión del simulador.

Los eventos generados por este componente del simulador tienen que estar definidos de forma que sean los suficientemente expresivos por sí mismos, y que no haya superposición

entre ellos, en cuanto a qué sucesos tienen cabida dentro de un tipo y de otro. Esto último es más complicado de conseguir. Sin embargo, siguiendo la clasificación de alto nivel que se hizo previamente (ver 2.3), se ha diseñado una familia de eventos que serán utilizados por el simulador, y habilitarán la comunicación entre módulos.

Este módulo actúa como una caja negra de cara al resto del simulador. No es necesario saber cómo ni por qué se generan los eventos, ni cuál es el proceso que hay detrás de dicha generación. Solamente será necesario que dicho módulo envíe un evento al módulo principal, y que ese evento pertenezca a las categorías anteriormente mencionadas.

Debido a la dificultad inherente al adecuado desarrollo e implementación de un módulo dedicado a la extracción de información de diversas fuentes, la implementación completa de este módulo sería demasiado costosa. Por este mismo motivo, y dada la flexibilidad que permite el diseño como caja negra, el desarrollo de la recogida y análisis de información, y su posterior transformación en eventos que puedan ser transmitidos al núcleo del simulador, se ha decidido retrasar para una versión posterior del simulador.

En esta versión se ha implementado una variante sencilla que generará un grupo reducido de eventos. Los eventos pertenecen a un tipo especial prototípico, al que se ha tenido a bien llamar DummyEvent. Son generados de forma periódica, y el simulador los recibe y los redirige al resto de módulos. Esto permite verificar el correcto funcionamiento de esta parte del simulador, y llevar a cabo el diseño e implementación del resto de módulo teniendo en cuenta que esta parte será ampliada en el futuro.

3.1.2. Módulo Principal (Controlador)

El módulo principal constituye el núcleo del flujo de información en el simulador. Es también, al mismo tiempo, el punto de inicio del simulador, quien inicia la ejecución cuando se lanza el sistema, y el encargado de establecer e inicializar el resto de módulos según sea necesario. Este módulo ha sido implementado en una sola clase llamada *MainClass*.

Arranque del módulo principal

Al ejecutar el simulador, este módulo se iniciará en un hilo, y comenzará la fase de arranque del resto de módulos requeridos para operar. Durante dicha fase, se suceden una serie de pasos que culminan con un simulador totalmente preparado para llevar a cabo las acciones que le sean requeridas:

1. Lo primero que hace el módulo es crear un hilo adicional desde el que realizará el resto de operaciones propias de su operación.

2. Después se arranca el módulo del flujo de eventos. Este módulo también es arrancado en su hilo propio, de forma que funcione independiente de los demás. Cuando este ha sido arrancado, el módulo principal es añadido a una lista de listeners en el flujo de eventos, de forma que recibirá todos aquellos eventos que el módulo de flujo de eventos vaya generando.

3. El tercer paso consiste en la creación del módulo de interfaz gráfica, y una vez más, delegar su ejecución a su propio hilo.

4. Como cuarto paso, se configura la interfaz gráfica con la información de los modelos de grafos de ataques que hay en el sistema. Esto permitirá a la interfaz gráfica disponer de una lista de modelos disponibles, que mostrará en pantalla para permitirle al usuario seleccionar aquel que quiera usar.

Una vez que se ha realizado la fase de inicialización, los módulos de flujo de eventos, el de la interfaz gráfica y el principal, se quedan a la espera, cada uno en un hilo de ejecución individual, para realizar sus tareas cuando se les solicite. Este enfoque permite que se ejecuten los tres módulos de forma independiente entre sí, y realicen sus tareas con la mayor concurrencia posible.

Ejecución continuada

Una vez inicializados los módulos básicos del simulador, este módulo principal se encargará de recibir solicitudes y redirigirlas a los módulos adecuados, y, hará también lo propio

con los eventos. Puesto que este módulo efectúa una función como controlador, las acciones que realice el usuario a través de la interfaz gráfica, serán dirigidas al módulo principal, quien se encargará de discernir cuál es su destino y realizar las solicitudes apropiadas a los módulos que correspondan.

3.1.3. Interfaz gráfica del simulador

El módulo de interfaz gráfica ha sido diseñado para ofrecer una visión general de todo lo que está aconteciendo en el simulador, así como permitir al usuario la realización de ciertas acciones. Puesto que uno de los objetivos fundamentales de nuestro simulador es ser capaz de proporcionar un modo de establecer una comparativa entre diferentes modelos de grafos de ataques, una de los cometidos principales de la interfaz gráfica debe de estar encaminado hacia la consecución de dicho objetivo.

Para lograrlo, la interfaz gráfica presenta una serie de gráficas durante la ejecución de los modelos, que permiten observar el número de nodos generado por el grafo de ataque, así como el uso de memoria y el tiempo de ejecución total. Además de lo anterior, también dispone de una ventana donde se ofrece un registro en el que se informa de todos los eventos recibidos por el módulo principal, y se da una breve descripción de ellos.

En la figura 6 podemos observar el aspecto de la interfaz gráfica principal del simulador. Cada uno de los elementos que se pueden observan en dicha interfaz son:

- En primer lugar, un panel con tres botones en la parte superior izquierda. Estos tres botones se utilizan, de izquierda a derecha respectivamente, para solicitar el inicio o parada del módulo de flujo de eventos, iniciar la ejecución del módulo de grafos de ataques (Decision Module), y para realizar el disparo de un evento de forma manual.

- Debajo de este, se puede observar el panel de información sobre los modelos de grafos de ataques disponibles. En este panel encontramos un menú desplegable que permitirá al usuario seleccionar uno de los modelos de grafos de ataques disponibles. Una vez seleccionado uno de ellos, veremos el nombre del modelo, debajo un campo de texto indicando quiénes son los autores, y finalmente el DOI del artículo en el que se

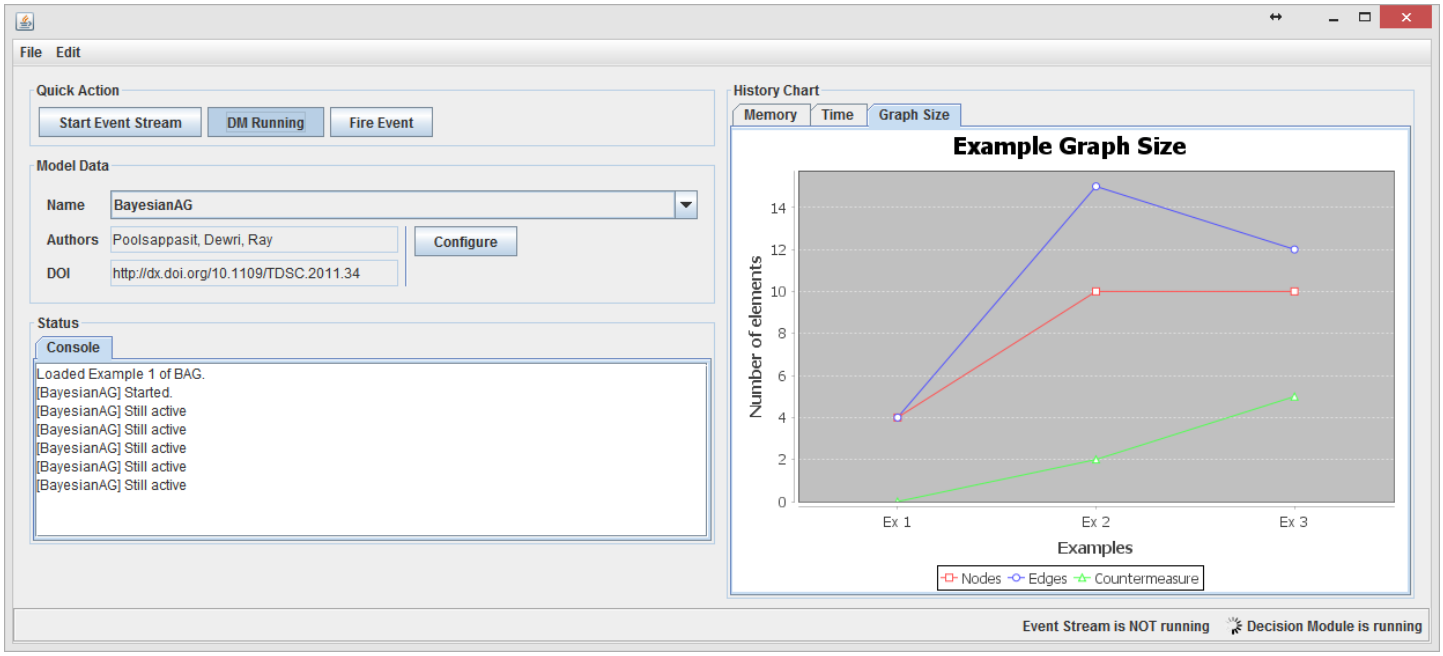

Figura 6: Ventana principal de la interfaz de usuario

publicó ese modelo, para poder obtener más información sobre el modelo en cuestión si así se desease. También se puede observar, junto al campo de los autores, un botón de configuración. Pulsando en ese botón, se abre una ventana de configuración para el modelo, proporcionada por el propio modelo.

- El panel que se observa en la parte izquierda de la interfaz, es una consola en la que se registra información adicional sobre el funcionamiento del simulador, en ella se podrán ver los eventos recibidos por el módulo principal y una descripción de los mismos, mensajes de estado proporcionados por los modelos durante su funcionamiento, y también los eventos de solución final que los modelos envíen al finalizar su ejecución.

- En la parte derecha se puede ver un panel con tres pestañas y una gráfica. Cada una de estas pestañas contiendrá una gráfica con información sobre la evolución de los grafos de ataques. La gráfica del tamaño de grafo, representará una evolución de los nodos, aristas y contramedidas existentes en el grafo a lo largo de su análisis por parte del módulo de decisión.

- La segunda pestaña contendrá una gráfica sobre tiempos de ejecución de los modelos en ejecución del simulador.
- La pestaña de la izquierda mostrará una gráfica del uso de memoria total que hace el programa en la ejecución del modelo.
- Por último, se ha incluido una barra de estado en la parte inferior que permita ver de un vistazo si los módulos de flujo de eventos están en ejecución, y también si lo están los de los modelos de los grafos de ataque.

3.1.4. Módulo de modelos de grafos de ataque

El módulo de modelos de grafos de ataques, es sin lugar a dudas el módulo más importante y complejo que se puede encontrar en el simulador. Es el encargado de permitir la inclusión de implementaciones de diversos modelos, para que puedan integrarse de forma adecuada en el simulador e interactuar con el resto de partes que componen a este sin la aparición de problemas inesperados. Debido a lo anterior, en el diseño de este módulo ha sido necesario estimar de forma adecuada cuáles serían las características que todo modelo de grafos de ataques tendrá, y cómo abordar las diferencias entre ellos.

El funcionamiento básico de un modelo implementado en este módulo consiste en lo siguiente: Recibirá eventos desde el módulo principal, y deberá reaccionar ante esos eventos. Realizará cálculos sobre su grafo de ataques, y proporcionará unos resultados y (posiblemente) una visualización de dichos resultados.

Sin embargo, no todos los tipos de eventos presentes en el simulador tienen por qué ser válidos para todos los modelos. Habrá modelos que solamente implementen un tipo, y los habrá que implementen varios. Esta posibilidad debe ser contemplada en el módulo. Un modelo proporcionará una lista de eventos soportados y reconocidos, de forma que la clase principal pueda solicitar dicha lista, y extraer de ella qué eventos redirigirle, y cuáles no.

Por otro lado, es imposible para el simulador conocer a priori qué características y propiedades debe contener y analizar cada modelo. La única información que ineludiblemente contendrán todos los modelos, es un grafo de ataques, con una serie de nodos y aristas. La solución diseñada para atajar este problema, es permitir a los modelos disponer de su

propia configuración específica sobre el grafo de ataques, y ofrecer una interfaz mediante la que puedan solicitar dicha información al administrador si no son capaces de obtenerla por sí mismos. Esto otorga la flexibilidad deseada al simulador, permitiendo la inclusión de cualquier modelo, mientras el mismo sepa cómo se le puede proporcionar la información que necesita para funcionar.

Interfaz del modelo

La implementación del modelo ha sido definida mediante un conjunto de clases que componen la base necesaria para que cualquier modelo sea integrado en el sistema. El núcleo de este módulo lo forma la interfaz *DecisionInterface*. Esta interfaz deberá implementarla cualquier modelo de grafos de ataques que se desee incluir en el simulador.

Esta interfaz define una serie de métodos que permiten al simulador llevar a cabo una interacción continua con el modelo, y obtener datos y resultados a través de la comunicación acontecida. La interfaz está formada por un total de 19 definiciones de métodos que se detallan a continuación:

- *getPaperName*: Este método se encarga simplemente de devolver una cadena con el nombre del artículo que se implementa en este modelo. Su función es puramente informativa de cara al usuario, y el resultado de una llamada a este método es el que se puede ver en la interfaz gráfica mostrada en la figura 6.

- *getPaperAuthors*: Al igual que el anterior, devuelve una cadena con el nombre de los autores del artículo implementado, con motivos informativos. Esta cadena se usa para informar al usuario en la interfaz gráfica principal.

- *getPaperDOI*: De una manera similar a los dos anteriores, este método se utiliza para obtener una URL con el DOI que permite buscar más información sobre el modelo de grafos de ataques que es implementado. Al igual que los dos anteriores se muestra en la interfaz gráfica principal al seleccionar el modelo correspondiente.

- *getRecognizedEvent*: Una llamada a este método devolverá una lista con los tipos de eventos que este modelo es capaz de reconocer y utilizar. Cada evento de uno de los tipos presentes en esta lista, serán redirigidos al modelo desde el simulador para que lo trate de forma adecuada.

- *getEnabledFeatures*: De un modo similar al anterior, se devolverá una lista de funcionalidades conocidas por el simulador, que un modelo puede o no implementar. Actualmente, la lista de funcionalidades posibles está formada por memoria, tiempo, y nodos. Correspondiéndose cada uno de esas funcionalidades con la capacidad de extraer información del modelo para representarla en las gráficas de la interfaz principal.

- *getNodesClass*: Este método de consulta proporciona información sobre el tipo concreto de nodo que utiliza este modelo. Los modelos pueden extender los nodos básicos proporcionados por el simulador para que éstos incluyan las características específicas del modelo en cuestión. La existencia de este método facilita al simulador la tarea de interactuar y realizar ciertas acciones con los nodos del modelo, aunque éstos hayan sido heredados y el simulador no los conozca a priori.

- *getEdgesClass*: Este método es análogo al anterior, excepto que en este caso se estará trabajando con las aristas del modelo. Devuelve la subclase específica que es utilizada por el modelo.

- *addNode*: Se utiliza para permitir al simulador añadir nodos al modelo. Solamente es necesario pasar un nodo del tipo adecuado como parámetro, y éste será incluido en el grafo de ataque del modelo. Puede que requiera algún tratamiento adicional por parte del modelo, para que el nodo contenga todas las propiedades necesarias.

- *addEdge*: Se utiliza para permitir al simulador añadir aristas al grafo del modelo. Es necesario pasar una arista del tipo adecuado como parámetro, y ésta será incluida en el grafo de ataque del modelo. Puede que requiera algún tratamiento adicional por parte del modelo, para que el arista contenga todas las propiedades necesarias.

- *delNode*: Éste método permitirá la eliminación de nodos del grafo de ataques. En este caso es necesario proporcionar el nodo que quiere eliminarse, y éste deberá ser un nodo

válido que pertenezca al modelo.

- *delEdge*: Análogo al anterior, facilita al simulador la eliminación de aristas del grafo del modelo.
- *getNodes*: Este método proporciona al simulador acceso a una colección en la que se encuentran todos los nodos del grafo de ataques que posee el modelo.
- *getEdges*: Al igual que el anterior, proporciona al simulador acceso a una colección. En esta ocasión la colección contendrá las aristas existentes en el modelo en el momento de solicitarla.
- *init*: Este método es el utilizado por el simulador para comenzar la ejecución del modelo de decisión implementado. En ella el modelo deberá iniciar todas sus estructuras y cargar los datos necesarios para empezar la simulación. Recibe como parámetro la clase del módulo principal, de forma que pueda agregarla como listener para los eventos de soluciones que el modelo deba devolver.
- *loadState*: Es un método que permite al modelo cargar su información de estado del grafo de ataques y sus estructuras, desde una base de datos proporcionada por el simulador. Durante la ejecución puede usar la misma conexión a base de datos aquí proporcionada para almacenar la información que necesite.
- *react*: El método mediante el cual los modelos que se implementen en el simulador reciben los eventos que han afirmado ser capaces de procesar. Los modelos deben reaccionar ante dichos eventos, y devolver un evento de solución al simulador.
- *getOverallRisk*: Este método constituye una forma rápida de conocer el riesgo que este modelo atribuye a la posibilidad de un ataque con éxito al nodo raíz. El riesgo se devuelve mediante un enumerado que puede tomar los valores *LOW*, *MEDIUM*, o *HIGH*.
- *getModelConfigurationFrame*: Este método solicita al modelo que proporcione información sobre la interfaz de configuración adicional, de forma que el simulador pueda ejecutarla y crearla para proporcionar al modelo la configuración adicional.

- *stop*: Este último método se utiliza para finalizar la ejecución del modelo, y eliminar el hilo de proceso correspondiente.

Todos estos métodos constituyen la interfaz de comunicación entre un modelo, y el resto del simulador. Una vez que un modelo haya implementado con la funcionalidad requerida cada uno de los métodos presentes en esta interfaz, estará listo para ser integrado en el simulador y ejecutarlo.

Ventana de configuración del modelo

Para la configuración adicional de las propiedades de un modelo, se ha diseñado una interfaz gráfica de configuración, que puede ser extendida fácilmente y permite la inclusión manual de nodos y aristas en el grafo de ataques, además de incluirles aquellas propiedades necesarias.

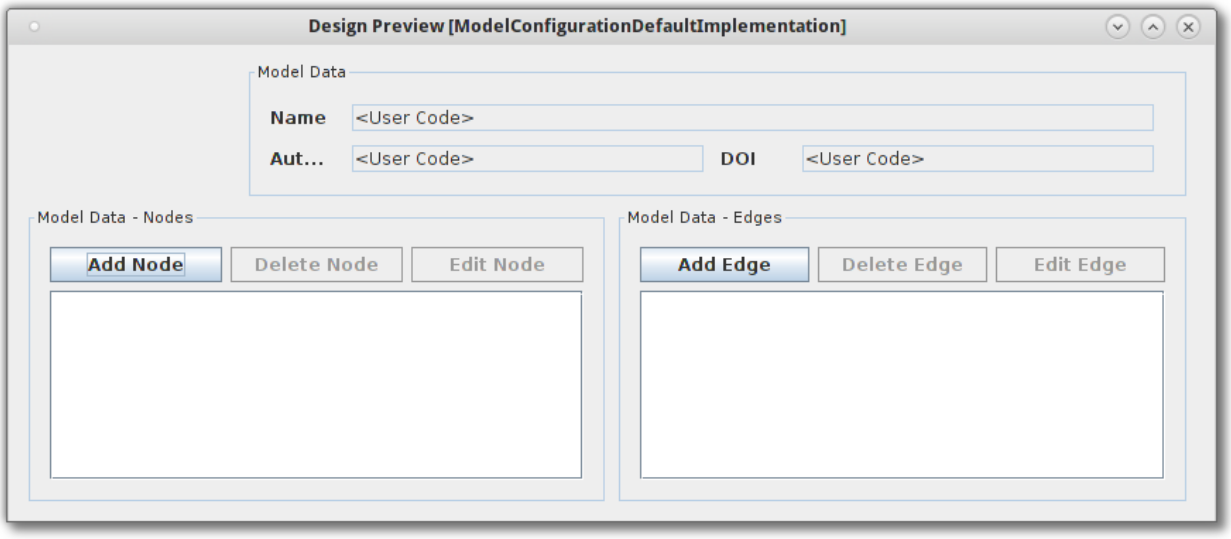

Figura 7: Ventana básica de configuración adicional del modelo

Se ha implementado una ventana de configuración básica, que se puede ver en la figura 7. La ventana ofrece la información básica del modelo, y una lista de nodos y aristas a las que se pueden añadir y de las que se pueden eliminar y editar dichos elementos. Al

solicitar la inclusión de un nuevo nodo o arista, o bien la edición de uno ya existente, se le presentará un diálogo de creación del respectivo componente del grafo, para especificar el nombre del componente, a qué otros nodos o aristas se une, etc. Este diálogo es ampliable de forma muy sencilla para incluir propiedades adicionales.

Características adicionales del modelo

Entre el conjunto de clases y herramientas proporcionadas para facilitar la inclusión de modelos al simulador, se incluyen, además de los mencionados con anterioridad, algunas características básicas adicionales. Estas son unas implementaciones básicas de nodos y aristas, que pretenden servir como un elemento básico y extensible para la creación de un modelo. Dichos nodos y aristas, vienen representados con un identificador, y una etiqueta. La etiqueta se ha pensado para que pueda utilizarse para describir propiedades de las aristas y nodos. También contienen estructuras y métodos para describir y trabajar con las relaciones básicas entre nodos y aristas en un grafo, tales como: unir dos nodos mediante una arista, obtener los nodos hijos de uno dado, obtener los nodos padre de uno dado, eliminar relaciones, etc.

No obstante, y como ya se ha visto en el resto de esta sección, toda la definición de funcionalidad de los modelos está diseñada para trabajar con subtipos de nodos y aristas, a los que se les pueden incluir los atributos y métodos adicionales que se deseen.

3.1.5. Visualización de grafos de ataque

El último de los módulos diseñados en el simulador, es el módulo de Visualización de los grafos de ataques. Este módulo está íntimamente relacionado con el de los modelos de decisión de grafos de ataques, ya que al fin y al cabo se trata de la visualización de estos. Los grafos de ataques son una técnica que se apoya de un modo bastante considerable en el elemento visual para el análisis de amenazas y vulnerabilidades en redes. Por lo tanto, a la hora de diseñar y desarrollar un simulador de diferentes modelos de análisis con grafos de ataque, no se puede dejar de lado la representación gráfica.

Una vez asentado el concepto de que era imprescindible mostrar una representación visual

de los grafos de ataques implementados como modelos de decisión, se decidió que la mejor forma de integrarlos en el simulador, era como un módulo independiente de los demás, solamente interrelacionado con los elementos proporcionados para los modelos. Al estar tan íntimamente ligado a la propia estructura interna de los modelos, era complicado conseguir que formase parte del módulo de la interfaz gráfica principal sin introducir acoplamiento.

Para la implementación de este módulo, se ha hecho uso de una biblioteca de grafos disponible para java, denominada *jgraphx*. Esta biblioteca permite una gran cantidad de personalización en el uso y representación de grafos. Sin embargo, en nuestro caso, en aras de ofrecer una interfaz lo más general y sencilla posible, hemos creado una clase fachada de esta librería, que permite generar grafos utilizando solamente la información ya disponible en los nodos y aristas básicos del módulo de modelos.

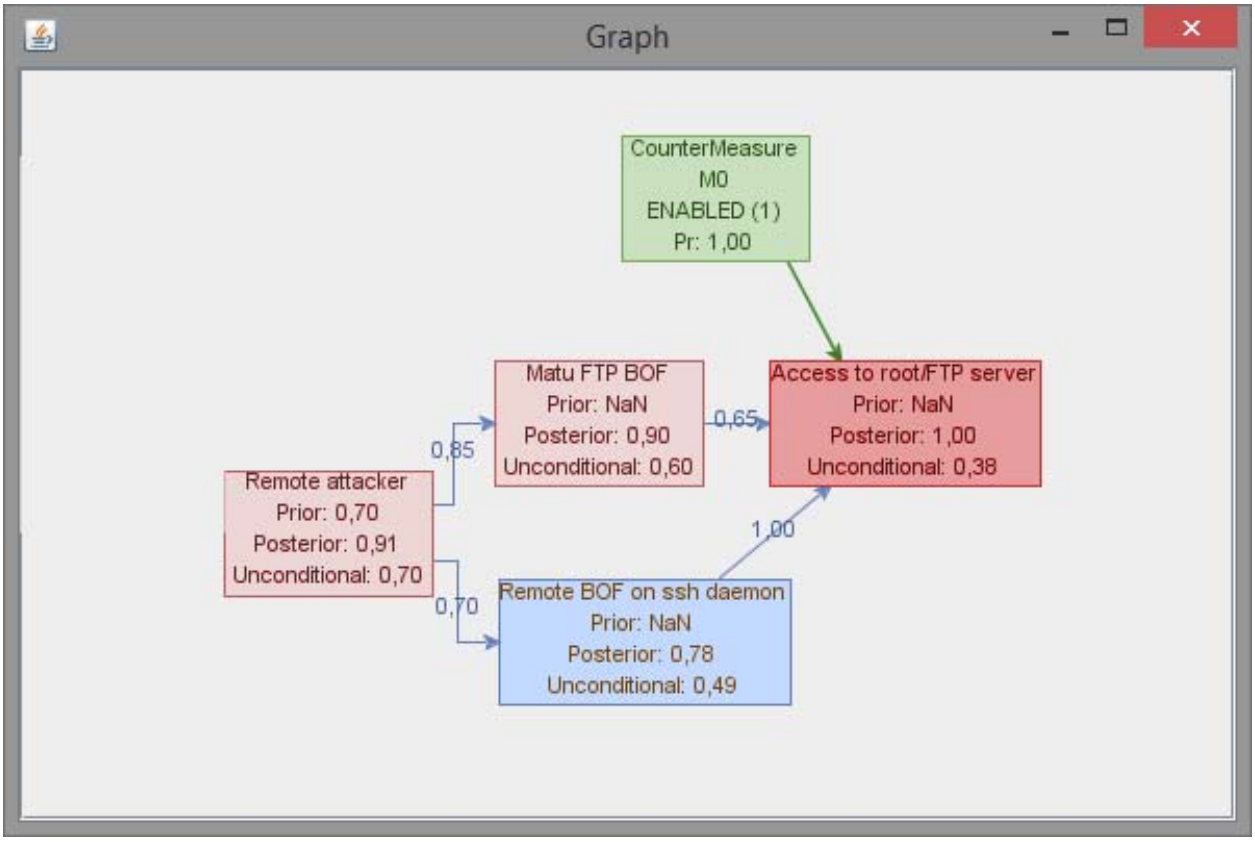

Figura 8: Ventana de visualización de los grafos del modelo

Con esto se consigue que los propios modelos implementados no tengan que preocuparse

de cómo se realiza la representación gráfica de su grafo, si no que solamente deben indicar que quieren que se añadan sus nodos y aristas a larepresentación. En la figura 8 se puede observar un ejemplo del funcionamiento de dicha representación. La contrapartida de esta interfaz simple y sencilla, es que se pierde control sobre ciertos detalles, como el posicionamiento de los elementos (se realiza automáticamente), o algunos detalles de estilo.

La interfaz que ofrece a los modelos la clase fachada generada es muy sencilla y se limita a operaciones simples:

- Añadir y eliminar elementos: Se pueden añadir y eliminar nodos de esta visualización con una simple llamada a la librería en la que se le pasa como parámetro el nodo o arista a añadir o eliminar. El módulo los añadirá, y mostrará una etiqueta que se corresponde con la proporcionada por el propio nodo o arista al solocitar su etiqueta.

- Modificar las etiquetas: En caso de que se quiera modificar la etiqueta por defecto de la visualización, el módulo permite realizarlo con una simple llamada en la que se incluye el grafo o arista a modificar, y la nueva etiqueta.

- Repintar el grafo: El módulo de visualización permite que se solicite repintar el grafo por completo. Esto es útil ya que el posicionamiento automático de elementos en el grafo no siempre realiza una colocación adecuada de los elementos, y el repintado puede servir para redibujarlo todo una vez que se completa la representación del grafo.

- Establecer el color de un nodo: El módulo de visualización permite realizar cambios en el color de relleno de nodos del grafo. Lo que resulta muy útil si se tienen diferentes tipos de nodos, o bien si queremos resaltar alguno en particular durante la ejecución del modelo.

- Establecer visibilidad de la visualización: Permite definir si se ve o no la ventana de la visualización gráfica. Es útil para cuando se quiere rellenar el grafo mientras éste no es visible, y mostrarlo una vez que la visualización contiene todos los elementos que debe tener.

Estas operaciones sobre el módulo de visualización son más que suficientes para realizar una representación gráfica del grafo de ataques del modelo simulado, y permiten al dichos

modelos abstraerse de la complejidad que conllevan las altísimas posibilidades de personalización de la biblioteca utilizada.

3.2. Modelos incluidos

En este apartado se explicarán con detalle cuáles han sido los modelos incluidos en el simulador, qué características tienen, y cómo se han implementado. Con el objeto de determinar si el simulador permite cumplir con las intenciones iniciales establecidas para este proyecto, se hacía preciso incluir al menos un modelo y comprobar que dicho modelo se puede integrar adecuadamente en el sistema, y que se pueden obtener resultados de él con los que alcanzar soluciones.

Finalmente, se ha decidido incluir dos modelos, con características diferentes entre sí, que permiten confirmar las decisiones de diseño adoptadas durante el desarrollo del proyecto. Los modelos de grafos de ataques incluidos son los detallados en los artículos siguientes:

- *Dynamic security risk management using bayesian attack graphs*[18]. Cuyos autores son Nayot Poolsappasit, Rinku Dewri e Indrajit Ray. El modelo propuesto en este artículo, recibe por parte de sus autores el nombre de *Bayesian Attack Graph*(BAG). Ese mismo nombre será el que se utilice en este trabajo para referirse a dicho modelo.

- *Scalable Optimal Countermeasure Selection using Implicit Enumeration on Attack Countermeasure Trees*[4]. Cuyos autores son Arpan Roy, Dong Seong Kim y Kishor S. Trivedi. El modelo propuesto en este artículo, recibe por sus autores el nombre de *Attack Countermeasure Tree*(ACT). Con ese mismo nombre se describirá el modelo correspondiente en este trabajo.

3.2.1. Bayesian Attack Graph

El modelo descrito en este artículo fue publicado en la revista *IEEE Transactions on Dependable and Secure Computing*, en el número de enero de 2012. En este artículo se propone un modelo de análisis de riesgos basado en el concepto de las redes bayesianas de

creencia, que, según indican los autores, permitiría a los administradores de redes cuantificar las posibilidades de que la red sea comprometida a varios niveles.

Resumen del artículo

Los autores parten de la noción de que la mayoría de modelos previos que hacen uso de grafos de ataques o árboles de ataques, no hacen consideración sobre las capacidades de un atacante, y por lo tanto, sobre la probabilidad de que ciertos ataques se lleven a cabo. Además, algunos modelos utilizan la lógica bayesiana para establecer las probabilidades de que un ataque tenga lugar. Pese a ello, en dichos artículos no especifican cómo calcular la probabilidad condicional de cada nodo del análisis. Otros modelos previos no permiten calcular un conjunto de soluciones óptimo, basado en la existencia de un presupueesto acotado. Finalmente se identifica una última cuestión sin resolver en trabajos previos, y es la naturaleza dinámica de las amenazas de seguridad, que hace que la probabilidad de sufrir un ataque específico, puede cambiar a lo largo de la vida de un sistema.

Las contribuciones que este artículo aporta para abordar estas limitaciones se resumen en:

1. Se propone el modelo BAG, que hace uso de redes de creencia bayesianas para contemplar ciertas condiciones de seguridad que afectan a un sistema comprometido. Es un modelo que incluye las características habituales de grafos de ataques en cuanto a causas y consecuencias de estados, y tiene en cuenta las probabilidades de explotación de esas características.

2. Se propone un método para estimar el riesgo de seguridad de una organización basado en la explotación de vulnerabilidades, haciendo uso para la estimación de las métricas CVSS.

3. Se establece un modelo para cuantificar el retorno sobre la inversión basado en las probabilidades de un compromiso de seguridad y un modelo de costes proporcionado por el administrador.

4. Se propone un algoritmo genérico para identificar planes de mitigación de riesgos óptimos, tanto en análisis con un objetivo único, como múltiples objetivos.

Funcionamiento del modelo

El modelo se basa en las redes bayesianas para elaborar sus grafos de ataques. Estas redes de probabilidad están constituidas por un grafo acíclico dirigido. Este modelo parte de unas definiciones de conceptos que son necesarias para entender el modelo.

- Plantilla de atributos: define los posibles atributos que pueden encontrarse en la red, pueden ser vulnerabilidades de sistemas, pivilegios de acceso, etc. Los atributos pueden ser asociados después con un estado (verdadero o falso) definiendo si ha ocurrido, y una probabilidad de que dicho estado sea verdadero. Dichos atributos serán los nodos del grafo.

- Ataque atómico: Formados por un conjunto de atributos de precondición, un conjunto de atributos de poscondición, y una probabilidad. Permiten a un atacante, si los atributos de precondición son verdaderos, obtener los estados de postcondición con la probabilidad definida.

- Grafo de ataques bayesano (BAG): Definido por S, un conjunto de atributos; A, el conjunto de ataques atómicos definido para dichos atributos; Un conjunto de tuplas de descomposición que permiten definir aristas con propiedades AND u OR; y un conjunto de funciones de distribución de probabilidad condicional, que permite representar la distribución local de probabilidad condicional de cada estado no inicial.

- Distribución local de probabilidad condicional (LCPD): Es una tabla que indica la probabilidad de que un atributo concreto sea comprometido, condicionado a los valores de los estados de los atributos precondición de éste.

Utilizar el modelo BAG para llevar a cabo un análisis de vulnerabilidades y establecer medidas para reducir estas, se hace siguiendo una serie de pasos. El primero de ellos consiste en estimar el riesgo en cada nodo. El administrador estima la probabilidad de que ocurran

los eventos de los nodos iniciales, mientras que los demás se introducen en las tablas LCPD en base a la dificultad de explotar un ataque atómico. Dicha dificultad se calcula mediante una fórmula en base a las métricas CVSS proporcionadas sobre una vulnerabilidad.

El siguiente paso, permite calcular la probabilidad incondicional sobre el nodo principal. Este cálculo se realiza mediante el teorema de bayes, utilizando las probabilidades de cada variable.

El modelo BAG incluye la capacidad de recalcular las probabilidades cuando se conoce evidencia de que un ataque ha tenido lugar, o por otros motivos. Una vez dichas condiciones cambian, se pueden establecer las probabilidades a posteriori para analizar el riesgo en la nueva situación. Esto se lleva a cabo calculando la probabilidad condicionada sobre aquellos atributos que ahora han sido comprometidos. Mediante una fórmula que permite aplicarlo sobre los atributos afectados, y haciendo uso nuevamente del teorema de Bayes, es aplicada sobre los nodos necesarios, y se obtienen las nuevas probabilidades.

Tras esto, se pasa a la fase de estudiar contramedidas. Para ello, el modelo BAG presenta lo que llaman controles de seguridad, que periten reducir la probabilidad de que uno o más atributos tengan lugar. Estos controles de seguridad, son un atributo por sí mismos. En este caso si están activos, disminuirán las probabilidades de ataque, y si no lo están, estas no sufrirán modificación alguna. Incluyen a qué nodos afecta su inclusión (qué nodos cubren), y además poseen un valor que representa el coste de implementación. La inclusión de los controles de seguridad producirá modificaciones sobre las tablas LCPD. El modelo permitirá además calcular un plan de mitigación, que indique aquellos controles de seguridad que deben implementarse como parte del proceso de mejora de la seguridad. La otra inclusión sobre la definición del grafo de ataques anterior, consiste en añadir a cada atributo un valor de potencial pérdida o ganancia, que serán utilizados en la decisión sobre medidas de seguridad.

Para elegir el conjunto de medidas a aplicar como solución, se desea reducir lo máximo posible el riesgo de ataque, con el menor coste de implementación. Para ello, este modelo presenta dos métodos de resolución.

- Optimización de un único objetivo (SOOP): En este metodo se pretende obtener aquel conjunto de contramedidas que minimice el coste y maximice la ganancia en seguridad, priorizando uno de estos objetivos sobre el otro mediante unos valores de pesos. La

elección adecuada de dichos valores depende del administrador, y puede ser complicada.

- Optimización de múltiples objetivos (MOOP): En este segundo método se eliminan los pesos, y se maximiza la ganancia de seguridad, mientras que se minimizan los costes todo lo que se pueda.

Ambos métodos se resuelven con el uso de un algoritmo genético que genera planes de seguridad y pretende obtener aquel que sea mejor. Aunque el método de selección de individuos es diferente entre uno y otro método. Cuando el algoritmo genético finaliza, el conjunto de medidas de control de seguridad que maximizan la mejora de seguridad es devuelto.

Implementación en el simulador

Para incluir el modelo Bayesian Attack Graph en el simulador, se han identificado aquellas características que el grafo subyacente tendrá, así como y que la implementación por defecto del módulo de modelos no proporciona.

Entre dichas características, se observa que existen dos tipos de nodos, por un lado el que representa ataques, por otro lado, el que representa contramedidas. Además, cada uno de ellos tiene atributos diferentes. Esto implica que han de implementarse dos tipos distintos de nodos para este modelo. Existe también, para los nodos de ataques, la tabla LCPD, las probabilidades a priori en el caso de los que son iniciales, el estado de dicho nodo (si ha tenido lugar el ataque, o si se ha implementado la contramedida), los valores de ganancias o pérdidas asociadas a la validez de dicho estado.

Una vez identificadas dichas características, y conocidas las estructuras adicionales necesarias a las del módulo base, se hace necesario también implementar los algoritmos que permitirán realizar los cálculos necesarios en cada paso de la ejecución del modelo. Para ello se ha establecido una función principal en el modelo, que implementa la interfaz *DecisionInterface*, y que además incorpora todos los algoritmos necesarios para llevar a cabo el análisis de riesgos y contramedidas proporcionado por este modelo.

Por último, puesto que este modelo incluye una cantidad importante de atributos que es necesario proporcionarle, se ha extendido la interfaz gráfica de configuración del modelo, para

que pueda solicitar todos los datos necesarios al usuario, y además proporcionarle alguna información adicional.

3.2.2. Attack Countermeasure Trees

El modelo descrito en este artículo fue publicado en el congreso *42nd AnnualIEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, de 2012. En este artículo se propone un modelo de análisis de riesgos basado en el concepto Árboles de contramedidas a ataques, que, según indican los autores, mediante el uso de diversos algoritmos para el análisis de varias funciones objetivo que minimicen las contramedidas necesarias para mejorar la seguridad al máximo.

Resumen del artículo

Este artículo parte de la noción de que el modelo formal básico de grafos de ataques, no tiene en cuenta contramedidas que se puedan implementar. En estudios anteriores los autores presentan el *Attack Countermeasure Tree*, que puede integrar contramedidas en cualquier nodo del árbol, permite generación y análisis automática de escenarios de ataque y escenarios de contramedidas, y facilitan la realización de análisis de seguridad. Así mismo, destacan que otros estudios similares adolecen del problema de explosión de estados, y presentan una solución para encontrar contramedidas de forma óptima.

Las contibuciones que este artículo realiza ante la visión de dichas limitaciones se resumen en:

1. Es posible realizar un análisis y optimización sin llevar a cabo asignaciones de probabilidades a los ataques, esto es relevante ya que existe un amplio sector que considera que es imposible obtener de forma adecuada probabilidades específicas para ataques.

2. Muestran como un conjunto de contramedidas óptimo puede ser seleccionado de todo el grupo de conjuntos de contramedidas posibles, utilizando un enfoque sin espacio de estados, lo que lo hace mucho más eficiente.

3. Se utilizan estrategias voraces y técnicas de enumeración implícita para computar en conjunto de contramedidas óptimo mediante varias funciones objetivo bajo diferentes tipos de restricciones.

4. Muestran la escalabilidad que las técnicas propuestas mantienen para grafos de gran tamaño.

Funcionamiento del modelo

El modelo se basa en el concepto de los Árboles de Contramedidas de Ataques. Dichos árboles pueden ser vistos como un grafo dirigido, donde un atacante parte de los nodos hoja para llegar hasta el nodo raíz, donde se culmina el ataque. Estos árboles tienen la particularidad de que para cada nodo de ataque, se puede establecer uno de contramedida que lo contrarreste.

Este modelo es muy versátil porque admite diferentes métodos para la selección de contramedidas. Métodos basados en probabilidad de que un ataque tenga lugar, o simplemennte orientados a evitar todos los vectores de ataque posible, asumiendo que no se conocen las probabilidades. Los métodos ofrecidos en este modelo se dividen en dos familias:

- Selección de contramedidas óptima sin asignaciones de probabilidad. Esta familia de técnicas permiten la selección de un conjunto de contramedidas sin contemplar probabilidades de que ocurran los ataques. Para trabajar sin probabilidades, este modelo se basa en el concepto de *mincuts*. Un mincut es aquel conjunto de nodos hoja imprescindibles para llevar a cabo el ataque, esto implica la presencia los nodos de ataques, y la ausencia de los de contramedidas que los anularían. El conjunto de todos los mincuts representa todas las vías de ataque posibles para alcanzar el objetivo.

  Dentro de esta familia, una de las técnicas propuestas permite seleccionar un número mínimo de contramedidas en el ACT. En primer lugar se establece un algoritmo que permite reducir el conjunto de todas las contramedidas especificadas en los mincuts, al mínimo posible para ofrecer una cobertura total del ataque. El resultado son aquellos mincuts necesarios cubrir el ataque completamente. No obstante, puede no ser posible cubrir el ataque completo (las contramedidas propuestas no lo permiten) o existir

restricciones de presupuesto que impidan la adopción de todas las contramedidas. Estos algoritmos suponen una cobertura parcial intencional (limitamos las contramedidas intencionadamente para cumplir el presupuesto) o no intencional (no es posible hacerlo aunque queramos). Estos dos casos se tratan como un caso especial del algoritmo de cobertura completa para su resolución.

La segunda técnica propuesta dentro de la familia sin probabilidades, toma en cuenta valores cuantitativos. Éstos son el coste de llevar a cabo los ataques (para el atacante), el coste de inversión en seguridad, y por último el cálculo del impacto. Con ellos se puede buscar el conjunto de contramedidas que permita maximizar el incremento de seguridad, minimizando el coste de inversión. Esto se lleva a cabo mediante un algoritmo de ramificación y poda.

- Selección de contramedidas óptima con asignaciones de probabilidad. Dentro de esta familia de técnicas, se implementan soluciones que permiten la selección de un conjunto de contramedidas atendiendo a las probabilidades proporcionadas para cada nodo del grafo. En primer lugar se presenta un algoritmo par ael cálculo de la probabilidad total del ataque. También se ofrecen métodos que permiten el cálculo del retorno de la inversión.

  Dentro de esta familia, se presenta también una técnica que permita maximizar el beneficio obtenido de la implementación de un conjunto de contramedidas, incrementando el nivel de seguridad con la menor inversión posible, al igual que se hacía en el caso anterior. Sin embargo, en esta situación, se lleva a cabo mediante un algoritmo que permite relacionar las probabilidades de que los ataques tengan lugar, aplicando diversas fórmulas sobre las probabilidades, y utilizando un algoritmo de ramificación y poda.

Implementación en el simulador

De igual manera a como se realizó con el otro modelo, para implementar este modelo dentro de nuestro simulador, ha sido necesario identificar aquellas características que será necesario incluir en el grafo subyacente, que no se encuentren ya en la implementación

por defecto.

En este tipo de grafo, disponemos de nodos que representan un ataque atómico, así como otros que representan una contramedida que mitiga dicho ataque. Dichos nodos pueden contener atributos adicionales como coste, probabilidad de suceso, impacto, etcétera. Será necesario por tanto incluir dos tipos de nodos nuevos.

Por otra parte, todas las diferentes técnicas se implementan de forma que se puedan ejecutar proporcionando todas la funcionalidad del modelo presentado. Esto se lleva a cabo en la clase principal del modelo, que también implementa la interfaz *DecisionInterface*.

Además, se extiende la interfaz de configuración, de forma que permita incluir los atributos que los elementos, solicitando al usuario los datos necesarios para ello.

En el desarrollo de este modelo, hemos encontrado una situación inesperada. El modelo presentado por los autores, realiza sus análisis en base al concepto de mincuts, sin embargo, no se proporciona un algoritmo o método para calcular cuales son estos conjuntos de mincuts que se derivan del grafo. En su lugar, se hace mención a una herramienta externa desarrollada por los autores, que se ha utilizado para obtener dichos conjuntos.

3.3. Resultados obtenidos

Finalmente,tras alcanzar el simulador concebido en este proyecto un estado de madurez mínimo que ha permitido el desarrollo y pruebas de los primeros modelos de grafos de ataques, se han podido obtener una serie de resultados sobre el simulador, y los modelos.

En primer lugar, se observa que la arquitectura implementada en el simulador es lo suficientemente versátil como para implementar modelos de manera satisfactoria, y hacerlos funcionar con la interacción adecuada del administrador, permitiéndole a este realizar pruebas sobre diferentes grafos de ataques y comparando los resultados.

Por otra parte, se observan en los modelos incluidos ciertas diferencias e incompatibilidades a la hora de poder comparar los resultados de estos. Los enfoques a la materia de los grafos de ataques, pueden llegar a ser tan diferentes entre sí, que la tarea de compararlos se complica por sus cualidades innatas. En este caso, se han probado dos maneras diferentes de abordar el problema que, a priori, parecían tener mucho en común y ser bastante parecidos

en cuanto a los datos y estructuras de partida, y al tratamiento que les daban: Ambos modelos utilizan una estructura de grafos muy similar, con el uso de nodos como explotaciones atómicas de una vulnerabilidad, la inclusión de nodos que pueden ser contramedidas, y el uso de atributos de probabilidad de que un ataque tenga lugar. Sin embargo pese a que todas esas premisas de partida son muy similares, la información que proporcionan de salida puede no ser capaz de compararse sencillamente.

Por último, se ha observado también que durante la inclusión de un modelo prometedor, se pueden encontrar carencias en la definición que se ha realizado de dicho modelo. Un enfoque que parecía estar bien definido y ser capaz de ofrecer una gran cantidad de técnicas de resolución, ajustando los resultados en base a la cantidad de información disponible, detalla de forma muy somera algunos de los procesos internos necesarios durante el análisis, y simplemente hace una referencia a una herramienta externa que, si tenemos suerte, podremos integrar y utilizar, pero también cabe la posibilidad de que ya no se encuentre disponible, sea propietaria, etcétera.

4. Conclusiones y vías futuras

En este trabajo se ha presentado un simulador para estudiar diferentes modelos de análisis y resolución de grafos de ataques, y establecer comparaciones entre ellos para observar sus diferencias y limitaciones. Las principales conclusiones obtenidas del desarrollo de este trabajo se detallan a continuación.

1. Se han estudiado en profundidad los paradigmas actuales de grafos de ataques, y los retos y limitaciones que estos encuentran en la actualidad.

2. Se ha hecho también un análisis sobre vulnerabilides, la base de datos de vulnerabilidades NVD, y cómo utilizar dichas vulnerabilidades para establecer precondiciones y postcondiciones para los grafos de ataques.

3. Se han analizado e incluido en el simulador dos modelos sobre grafos de ataques, obteniendo unos resultados preliminares bastante interesantes. Con la experiencia obtenida durante el desarrollo del simulador y las pruebas iniciales de modelos, se ha podido observar que debido a la heterogeneidad que existe entre estos modelos, no es en absoluto fácil establecer un marco que permita comparar de forma adecuada diferentes modelos de grafos de ataques.

4. Se han diseñado e implementado unas interfaces que permiten la programación e integración de nuevos modelos en el simulador, facilitando la extensión futura de este, y ampliando su utilidad.

5. También se ha observado que un simulador de este tipo permite identificar aquellos conceptos que no se han definido del todo bien en modelos de grafos de ataques, y estudiar alternativas a estos problemas.

Esta primera versión del simulador presenta una herramienta prometedora, novedosa y con un alto potencial de servir de utilidad a administradores de redes.

Entre las vías abiertas para el desarrollo en el futuro de este simulador, se pueden destacar:

1. Integración de los eventos para favorecer la automatización de las simulaciones. Se podría hacer a través de dos vías, independientes entre sí e interrelacionadas: Integración de eventos que faciliten información sobre el estado de la red, su topología y configuraciones, permitiendo al simulador y a los modelos modificar los grafos según varíe dicha información; eventos sobre vulnerabilidades recolectados desde bases de datos de vulnerabilidades.

2. Estudiar un posible marco o métrica de comparación de modelos de grafos de ataques, que establezca unos criterios de comparación adecuados.

3. Ampliar el módulo de visualización de grafos, de forma que se puedan incluir técnicas como la agregación o reducción si un modelo lo solicitase, facilitando la visualización de grafos de ataques de gran tamaño.

4. Ampliar el módulo de visualización, para ofrecer una mayor interactividad con el administrador, que podría usar el ratón sobre él y obtener información adicional de los nodos, editar sus propiedades, etc.

5. Por supuesto, incluir nuevos modelos de grafos de ataques. Cuantos más modelos diferentes se puedan integrar en el sistema, mayor será la utilidad de este.

Referencias

[1] Charles P Pfleeger and Shari Lawrence Pfleeger. *Security in computing*. Prentice Hall Professional Technical Reference, 2002.

[2] Richard Paul Lippmann and Kyle William Ingols. An annotated review of past papers on attack graphs. Technical report, DTIC Document, 2005.

[3] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[4] A. Roy, D.S. Kim, and K.S. Trivedi. Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees. In *Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 1–12, June 2012.

[5] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284. IEEE, 2002.

[6] Arpan Roy, Dong Seong Kim, and Kishor S Trivedi. Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees. In *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*, pages 1–12. IEEE, 2012.

[7] Oleg Sheyner and Jeannette Wing. Tools for generating and analyzing attack graphs. In *Formal methods for components and objects*, pages 344–371. Springer, 2004.

[8] CR Ramakrishnan, RC Sekar, et al. Model-based analysis of configuration vulnerabilities. *Journal of Computer Security*, 10(1/2):189–209, 2002.

[9] Xinming Ou, Wayne F Boyer, and Miles A McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 336–345. ACM, 2006.

[10] Ghanshyam S Bopche and Babu M Mehtre. Attack graph generation, visualization and analysis: Issues and challenges. In *Security in Computing and Communications*, pages 379–390. Springer, 2014.

[11] Sushil Jajodia, Steven Noel, and Brian O'Berry. Topological analysis of network attack vulnerability. In *Managing Cyber Threats*, pages 247–266. Springer, 2005.

[12] Todd Heberlein, Matt Bishop, Ebrima Ceesay, Melissa Danforth, CG Senthilkumar, and Tye Stallard. A taxonomy for comparing attack-graph approaches. *Online] http://netsq. com/Documents/AttackGraphPaper. pdf*, 2012.

[13] Steven Noel and Sushil Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 109–118. ACM, 2004.

[14] Mohammed Alhomidi, Martin J Reed, et al. Attack graphs representations. In *Computer Science and Electronic Engineering Conference (CEEC), 2012 4th*, pages 83–88. IEEE, 2012.

[15] CERT. Vulnerability analysis. `http://www.cert.org/vulnerability-analysis/`. Accessed: 2015-09-04.

[16] NVD. National vulnerability database. `https://nvd.nist.gov`. Accessed: 2015-09-04.

[17] Sebastian Roschke, Feng Cheng, Robert Schuppenies, and Christoph Meinel. Towards unifying vulnerability information for attack graph construction. In *Information security*, pages 218–233. Springer, 2009.

[18] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using bayesian attack graphs. *Dependable and Secure Computing, IEEE Transactions on*, 9(1):61–74, 2012.