Paulo Gaona-García*
Carlos Montenegro-Marín**
Julio Barón Velandia**
Universidad Distrital Francisco José de Caldas, Colombia

Modelo ontológico para la predicción de ataques informáticos a partir de *Honeynets* virtualizadas

Ontological model for predicting cyberattacks based on virtualized *Honeynets*

Modelo ontológico para a predição de ataques a computadores de Honeynets virtualizados

Resumen

Las *honeynets* son herramientas de seguridad informáticas usadas con el propósito de reunir información de posibles atacantes acerca de las vulnerabilidades presentes en la red. Para realizar un correcto uso de ellas es necesario entender los tipos existentes, las estructuras planteadas, las herramientas usadas y los avances actuales. Sin embargo, una mala planificación de un *honeypot* o *honeynet* podría proveer a usuarios indeseados un punto de acceso a la red que deseamos proteger. El propósito del siguiente artículo es llevar a cabo el planteamiento de un modelo ontológico para la identificación de tipos de ataques más comunes a partir del uso de *honeynets*, y su implementación sobre escenarios de trabajo. Este modelo facilitará la toma de decisiones para la ubicación de elementos y componentes a nivel informático en una organización.

Palabras clave: ataques informáticos, modelo ontológico, seguridad, vulnerabilidad.

Abstract

The *honeynets* security tools are widely used today for the purpose of gathering information from potential attackers about vulnerabilities in our network. For performing correct use of them is necessary to understand the existing types, structures raised, the tools used and current developments. However, poor planning *honeypot* or *honeynet* one could provide unwanted users an access point to the network we want to protect. The purpose of this article is to carry out the approach of an ontological model for identifying the most common attacks types from the use of

Fecha de recepción del artículo: 14 de junio de 2016
Fecha de aceptación del artículo: 28 de agosto de 2016
DOI: http://dx.doi.org/10.22335/rlct.v8i1.344

*Doctor en informática, Facultad de Ingenieria, Docente investigador. Universidad Distrital Francisco José de Caldas. Contacto: pagaonag@udistrital.edu.co
**Doctor en Sistemas y Servicios Informáticos para Internet, Universidad de Oviedo. Docente de la Facultad de Ingeniería, Universidad Distrital Francisco José de Caldas, Bogotá-Colombia. Contacto: cemontengrom@udistrital.edu.co
***Doctorado en Ingeniería Informática. Universidad Pontificia de Salamanca Campus de Madrid. Docente investigador. Contacto: Universidad Distrital Francisco José de Caldas. Contacto: jbaronv@udistrital.edu.co

honeynets, and its implementation on working scenarios. This model will facilitate decision-making for the location of elements and components to computer level in an organization.

Keywords: computer attacks, ontological model, vulnerabilities, security.

Introducción

La seguridad en las redes de computación será siempre un tema relevante y de interés a nivel informático. Desde la invención y masificación de Internet, la cantidad de usuarios y la información personal almacenada creció de forma exponencial, por lo tanto, garantizar la seguridad de este tipo de información llevó a una carrera de innovación de herramientas no solo para brindar seguridad, sino también para traspasarla (Romney et ál., 2005). Es así como las *honeynet* se consideran como una de las herramientas más usadas para probar fallos de seguridad en una red (Zurutuza et ál., 2011).

Un concepto claro de *Honeynet* podemos encontrarlo en (Sqalli et ál., 2011), donde los autores la definen como una solución cuyo objetivo es reunir información acerca de amenazas de seguridad que presente nuestra red, para de esta forma mejorar sus debilidades (Watson & Ridden, 2008). El concepto de *honeynet* es sobre el que estará enfocado el desarrollo del artículo, pero para entenderlo es necesario ubicarnos en la base sobre la cual se hizo todo este desarrollo. De acuerdo con Gallego y López (2004) se presentan enfoques y definiciones claras y simples acerca de los elementos contenidos en las *honeynets*, donde se puede resaltar qué es una *Honeynet*, sus arquitecturas tanto las de primera como de segunda generación, la realización de *honeynets* virtuales así como las herramientas de virtualización que llevan a cabo esto último.

Por su parte una *honeypo*t de acuerdo a (Memari et ál., 2014) es una herramienta que se utiliza como señuelo, con el fin de ser atacada, por esto todo tráfico que pase por ella será detectado como un ataque o intrusión. Una *honeypot* no tendrá comunicación alguna con ninguna entidad y, en caso de detectar tráfico cuyo origen sea el *honeypot*, este será tomado inmediatamente como una amenaza. Sin embargo, existen algunos inconvenientes en el uso de este tipo de herramientas (Vergel, Martínez, Zafra, 2016). En el trabajo realizado por Karthik et ál. (2009) presentan de manera explícita el riesgo que se corre al implementar un *honeypot* en entornos académicos. Uno de los inconvenientes que resaltan los autores, es cuando se requiere una instalación con total precisión y cuidado, de tal forma que la seguridad de la red no se vea comprometida en este punto, además de otros problemas legales que esto acarrearía.

Las *honeynets* son *honeypot*s más potentes y complejas dado que permiten un nivel mayor de interacción con los atacantes de la red. Las *honeynets* se encuentran dentro una red implementada y lista para ser atacada por lo que permite una mayor cantidad de información sobre los sistemas atacantes, comúnmente realizadas a partir de varias *honeypot*s. Propuestas realizadas por (*Chang & Tsai*, 2010) presentan ejemplos gráficos de la forma de realizar las *honeynets*; como resultado de esta actividad, se puede proponer la colaboración de sistemas *honeynets* virtuales (VHS), para mejorar los diseños y conceptos de las arquitecturas *honeynets*.

A partir de estos referentes, el presente artículo tiene como propósito llevar a cabo la definición de un modelo ontológico que facilite la identificación de ataques informáticos en un entorno de *Honeynets* virtualizadas, con el propósito de definir un *framework* de visualización de ataques informáticos para el análisis de comportamientos de vulnerabilidades sobre entornos corporativos.

Estado del arte

La educación sobre seguridad de datos se ve altamente beneficiada por el análisis de datos *honeynet*. En el estudio realizado sobre educación para el análisis de datos *honeynets* por (Romney et ál., 2005; Vergel, et.al, 2016) presenta el concepto de las universidades como entornos atractivos para la generación de una red trampa mediante un caso de estudio realizado en la instalación de un

ISSN 2145–549X | ISSN 2422-4200, Vol. 8, No. 1, Julio - Diciembre 2016

laboratorio de seguridad en la Universidad *Brigham Young* (BYU). El laboratorio ofrecía un entorno seguro (Carreño, 2014) para configurar y supervisar las diferentes herramientas provistas por el proyecto *Honeynet*.

El proyecto *Honeynet* es una organización dedicada a la investigación de seguridad computacional. La meta del proyecto es "aprender sobre las herramientas, las tácticas y motivos envueltos en los ataques a redes de computadoras" (Watson & Ridden, 2008). De acuerdo con Watson & Ridden (2008) se muestran las herramientas usadas para la recolección de datos en *honeynet* aunque solo muestran las más usadas por el proyecto *honeynet*. A continuación, en la tabla 1, se presentan los tipos de herramientas e interacción que se generan y sus características principales.

Tabla 1

Herramientas *Honeynet*

Herramienta	**Tipo de *honeypot***	**Característica principal**
Honeyned	Baja interacción	Emula los almacenes de IP de varios sistemas operativos y provee de manera opcional servicio de emulación básica.
Nephentes	Baja interacción	Emula vulnerabilidades comunes de MS Windows; es extremadamente bueno capturando *malware* como gusanos.
Honeytrap	Baja interacción	Detecta intentos de conexión contra puertos TCP no conectados.
Kojoney	Baja interacción	Emula los procesos de un servidor SSH y graba los nombres de usuario y contraseñas de los atacantes.
Sebek	Alta interacción	Monitorea *honeypot*s de alta interacción, engancha llamadas de lectura o escritura para capturar el acceso de archivos o actividad de entrada y salida.
Hflow	Herramienta de fusión para análisis de *honeynet*	Cruza datos obtenidos de *Snort, pof y sebek* en una estructura de datos relacionada para almacenamiento en una base de datos relacional.
Honeywall	CDroom iniciable	Es usado para construir de forma rápida una *honeynet* de alta interacción. Permite un análisis y control transparente de los datos.
Capture HPC	Alta interacción	Ejecuta aplicaciones dentro de una máquina virtual de Windows, aunque su mayor uso ha sido encontrar posibles URL maliciosas.
S*pamPot*	*email*	Colecciona y analiza mensajes vía email.

Fuente: Autores

En los estudios relacionados sobre el uso de una red trampa, se presentan las herramientas *honeynets*. Estas implementan sistemas de detección de intrusos los cuales generalmente manejan sensores virtuales con los cuales se puede recibir todo el tráfico de la red, para posteriormente ser analizado como un intruso o sospechoso por defecto. Estos IDS implementados en la *honeynet* presentan bastante información, la cual permitirá un análisis profundo del tráfico para posteriormente realizar una detección en un registro de todos los paquetes que llegan a los sistemas *honeynets*.

Las *honeynets* también pueden ayudar en la eliminación de acceso a los sistemas reales, por lo que representan una emulación física de los

sistemas y servicios en la red, retrasando así al intruso como lo presentan los autores (Memari et ál., 2014). Se detalla una visión general de una implementación basada en un contenedor ligero que emula sistemas operativos populares como *Linux* y *Windows* y proporcionan servicios a los intrusos desprevenidos. Los resultados muestran los ataques del mundo real en contra del sistema implementado, con datos como qué protocolo fue el más usado para realizar los ataques y desde qué zona geográfica fueron hechos los ataques.

En un estudio realizado sobre integración de un sistema *honeynet* con *botnet* (Yu et ál., 2009), se presentaron las diferentes integraciones de sistemas *honeynets* con otros modelos y sistemas de seguridad como lo son los *botnets*. Un *botnet* es un robot informático autónomo que puede controlar servidores u ordenadores de forma remota. La utilización de sistemas informáticos, como robots informáticos, llevan una cantidad mayor de información sobre los ataques de la red por lo que presentaron su implementación. La integración de estas técnicas de seguridad, ofrece un sistema de detección de tasas tanto para falsos positivos o negativos, y para ayudar a los sistemas de control a prevenir los ataques de gusanos informáticos, liberando en tiempo real gusanos benignos que contraataquen el gusano malicioso.

La arquitectura *bot-honeynet* puede no solamente detectar ataques maliciosos de gusanos sino también defenderse de ellos, haciendo una combinación de filtros, sistemas de detección de anomalías y *honeynet*, de forma que se realice acentuando las fortalezas y camuflando las debilidades de este sistema.

En el trabajo de Yao et ál. (2009) se presentan diferentes diagramas que muestran la arquitectura, compuesta principalmente de un controlador y un número de *honeypots*, donde cada *honeypot* está compuesto de un filtro, un sensor de detección y un *honeypot*. De acuerdo a este modelo se presenta un controlador usado por los *honeypots* del sistema de forma manual o automático. Cuando el mensaje entrante de algún *honeypot* es actualizar las reglas del filtro del sensor de anomalías, la orden se realiza automáticamente. Cada *honeypot* contiene un componente de filtro el cual es usado para bloquear ataques de gusanos conocidos. Una vez detectado el gusano, el filtro notifica al controlador para que este ordene a los *honeypots* desplegar el gusano benigno, para realizar el contraataque. Entre otros usos, las redes *honeynets* también se pueden implementar para mejorar el tiempo de respuesta frente a ataques DDoS como lo presentan (*Kwon* et ál., 2012) donde la problemática de los ataques DDoS o ataques de denegación de servicio, son ataques informáticos a un servicio o recurso, mediante el colapso de la red de banda ancha o una sobrecarga a los recursos del sistema.

Metodología de trabajo

Definición de un modelo ontológico

La web semántica es una extensión de la *World Wide Web* (www) actual, que permite mejorar las conexiones entre computadoras para realizar una cooperación entre personas. Las ontologías son definidas por (Uschold & Gruninger, 1996) como una definición formal y explícita de una conceptualización compartida de un dominio de particular interés. Lo que pretende realizar la ontología, es generar un conjunto de axiomas que generan una serie de afirmaciones lógicas y para las cuales son necesarias la utilización de tres elementos: las personas, las clases y las propiedades que, por medio de un software al que se llama razonador, se pueden identificar leyes ocultas sobre un escenario planificado.

A través del tiempo el desarrollo de ontologías ha tenido una evolución constante, autores como (Yu et ál., 2011), muestran las ventajas del desarrollo de ontologías, las cuales pueden resumirse en compartir conocimiento e información de forma más precisa, puesto que las ontologías son usadas para representar información en cualquier campo de conocimiento, donde cada concepto tiene atributos propios y a partir de la correlación de estos, es más fácil identificar diferencias y similitudes.

La ontología es un conjunto de axiomas que proporcionan afirmaciones lógicas explícitas, acerca de tres tipos de cosas: las clases, los

ISSN 2145–549X | ISSN 2422-4200, Vol. 8, No. 1, Julio - Diciembre 2016

individuos y las propiedades. A través de un software razonador, se pueden inferir otros hechos que están contenidos implícitamente en la ontología. El software razonador seleccionado para la realización del modelo ontológico es *Protégé* (Noy et ál., 2003) ya que los métodos de visualización tienen una gama de características diferentes. De acuerdo a (Katifori Akrivi et ál., 2006) *protégé* se presenta como una herramienta ampliamente utilizada en la creación de ontologías y su entorno de código abierto presenta muchas posibilidades de mejora o creación de nuevas funcionalidades en forma de *plugins*. Para su manejo, se utilizó un manual realizado por (Horridge, 2011), intuitivo y de fácil comprensión.

Datos utilizados

Para la definición de los conceptos utilizados en la ontología se tomó como referencia la OWASP (*Open Web Application Security Project*), organismo sin ánimo de lucro que se encarga de determinar y combatir los ataques más frecuentes de aplicaciones web. Basados en la recopilación de información sobre ataques web, la fundación presenta cada tres años un top 10, donde se exponen los ataques más frecuentes y con mayor peligrosidad de la red, por lo que es una gran herramienta de consulta. (Papapanagiotou & Spyros, 2013). Teniendo en cuenta que existen varias fuentes de datos abiertos para reportes de datos de seguridad informática, se utilizó esta fuente por:

- Su imparcialidad. Al ser un proyecto libre, no tiene presiones corporativas y entrega información imparcial, práctica y redituable.
- Su actualidad. La comunidad del proyecto OWASP está conformada por empresas, organizaciones educativas y particulares de todo mundo, por lo que siempre se encuentra actualizada.
- El grupo global de voluntarios de la OWASP supera los 45.000 miembros, atentos a resolver dudas que se tengan con respecto al proyecto.

Tipo de ataques más comunes

Uno de los proyectos bandera, de la OWASP, es el top 10 de vulnerabilidades más comunes y de mayor riesgo en aplicaciones web. Este top se actualiza cada 3 años y su última actualización es del año 2013. Basándonos en el top 10 de la OWASP se realizó la presentación de tres escenarios con los ataques más importantes del top para el año 2013, los cuales, dada su relevancia, hay que aprender a identificar y prevenir por parte de los administradores de redes y desarrolladores de aplicaciones web.

SQL injection

Es uno de los problemas más comunes al momento de realizar aplicaciones de seguridad en la web. El atacante envía ataques de texto simple, los cuales se enfocan en explotar la sintaxis del intérprete; esto ocurre en aplicaciones que presentan filtración inadecuada de variables de entrada. A través de este ataque se podrían leer todos los registros privados, debido a la alta flexibilidad de los lenguajes SQL.

(Sadeghian et ál., 2013) introduce el concepto de los ataques, los cuales se desarrollan muchas veces debido a falta de experiencia en el desarrollo de aplicativos SQL dejando vulnerabilidades en el aplicativo al momento de hacer búsquedas dinámicas a través de la concatenación de declaraciones con variables; los atacantes buscan obtener acceso a un parámetro que la aplicación web permite. (Sadeghian et ál., 2013) proponen una solución única puesto que el lenguaje SQL es muy flexible, aunque plantean una combinación de dos métodos conocidos para lograr un mayor nivel de seguridad:

a. La implementación de búsquedas parametrizadas. El motor de búsqueda SQL primero hará un barrido y compilará la petición sin las variables, manteniendo el resultado; posteriormente,

añadirá las variables y realizará el proceso de compilación de nuevo, de tal forma que las búsquedas maliciosas sean tratadas como cadenas de datos ordinarias.

b. Una configuración inteligente del motor de administración de la base de datos. A partir de la definición de diferentes usuarios y la limitación de sus privilegios, para que al momento de encontrar un fallo, el atacante únicamente pueda ver la información y no borrarla (Ma et ál., 2011). Nos muestra una solución a este tipo de ataque por medio de la implementación de un *honeypot* y cómo este último intercepta las sentencias SQL maliciosas.

Session management and broken authentication

Debido al número de servicios ofrecidos por la web tales como transacciones monetarias e intercambio de información (Huluka & Popov, 2012) listan las causas de secuestro de sesiones, tales como:

- El uso de una sesión de identificación adivinable.
- Ausencia de mecanismo de detección de pruebas de adivinación de sesión.
- Inhabilidad de detectar intentos repetidos de adivinar.
- Criptografía débil en el algoritmo o una debilidad en la forma de uso de un algoritmo de criptografía seguro.
- Manejo inseguro de métodos de sesión.
- Y las causas básicas de ruptura de autentificación:
- Falta de métricas: ausencia de métricas correctamente desarrolladas para la toma de decisiones en cuanto a la elección de mecanismos de seguridad.
- Falta de conocimiento de los programadores para aplicar mecanismos de seguridad de comunicación e información.
- Uso de módulos autodesarrollados en contraposición de módulos confiables previamente diseñados y probados.
- El nivel de información de un usuario en el sistema.

Por su parte (Wazzan & Mohammad, 2015) presentan una tabla con una particular comparación de los tipos de ataque que se presentan en la red. La tabla 2, presenta varios tipos de ataque mostrando el tipo de ataque, por qué solicitud se realiza el envío y el evento que genera cada ataque.

Tabla 2

Tipos de ataques a redes informáticas

Aspecto de la solicitud	Tipo de ataque	Evento
URL	Búsqueda energética	Pedir referencias URL que no están definidas como puntos de acceso, o no existen.
	Sospecha de violación de acceso	La URL contiene metacaracteres no permitidos o el tiempo de acceso válido expiró.
	Filtración de información	Usando métodos ilegales de HTTP en la petición o no permitidos en el método SOAP.

ISSN 2145–549X | ISSN 2422-4200, Vol. 8, No. 1, Julio - Diciembre 2016

Método	Violación RFC	La declaración *GET o HEAD* contiene un cuerpo o body.
Host	No hay cliente de navegación	No existe encabezado de *host* en la petición HTTP 1.1.
	No hay cliente de navegación	El encabezado del *host* contiene la dirección IP.
Cookies	Ataque de contrabando de petición http.	El encabezado de la cookie no es RFC obediente.
URL referencia	Ataque automatizado	La referencia del encabezado contiene una referencia URL no identificada.
Datos sensibles	Filtración de información	La respuesta contiene datos sensibles tales como información crediticia.
Archivos sospechosos	Detección de virus	La petición contiene un archivo que contiene un virus o un gusano.
Patrón sospechoso	Firma de ataque	La petición o respuesta contiene un patrón idéntico a una firma de ataque.
IP	Ataque DOS	La petición contiene una IP no confiable o en lista negra.

Fuente: autores.

La tabla 2 presenta un resumen de algunos de los ataques más utilizados para lograr acceder a una red y vulnerar la seguridad con la cual fue implementada. Existen una gran cantidad de ataques, sin embargo, se consideran estos como los más comunes y representativos.

Cross-site scripting (XSS)

(Gupta et ál., 2015) define el *cross-site scripting* como una vulnerabilidad de tipo inyección de código a nivel de aplicación, la cual ocurre cuando un servidor o página web usa entradas sin restricción vía HTTP, *database* o archivos en su respuesta sin ninguna validación, lo cual permite a un usuario malicioso hurtar información sensible.

Estas vulnerabilidades son de tres tipos: almacenamiento, reflejadas y DOM *based*. Las primeras ocurren cuando la entrada de un usuario se almacena en la base de datos y posteriormente es usada en la página de respuesta. Las reflejadas ocurren cuando la entrada de un usuario es referenciada en una página web de respuesta inmediata, sin la apropiada validación. Las vulnerabilidades DOM-*based* ocurren cuando un programa externo del cliente usa entradas de usuario no válidas obtenidas de forma dinámica de la estructura modelo de documentos de objeto.

En cuanto al enfoque del trabajo sobre *honeypot*s (Djanali et ál., 2014), proponen un método en el cual el *honeypot* no solo registre el ataque, sino que realice un apropiado contraataque al exponer la identidad del atacante; en él se evidencia todo el proceso de diseño del *honeypot* por medio de código *Javascript*. En caso de una petición normal, el *honeypot* proveerá una respuesta normal, en caso contrario, el *honeypot* identificará el ataque y enviará una respuesta como si el ataque hubiera sido exitoso; posteriormente se insertará código *javascript* en la respuesta, el cual será ejecutado por el buscador del *cracker* y recolectará información para ser enviada de nuevo al *honeypot*.

Fases del modelo ontológico

Una vez identificados los tres ataques con los que trabajamos, lo que se hizo a continuación es plantear el modelo ontológico de acuerdo a la parametrización de estos escenarios.

Definición de entidades que hacen parte de la ontología

Figura 1

Entidades que hacen parte del modelo ontológico. Fuente: OWASP *Threat Agent* (OWASP, 2011.

Para implementar las clases del modelo ontológico tomamos como modelo de referencia la ***OWASP Risk Rating Methodology,*** donde los riesgos vienen dados por el modelo matemático presentado en 6 pasos diferentes en la siguiente ecuación, tal como se presenta en la figura 1 y se ha representado en la ecuación 1.

$$Riesgo = Probabilidad * Impacto$$

Ecuación 1. Modelo estándar. *OWASP Threat Agent (OWASP, 2011)*

Los pasos descritos en la figura 1 dan lugar a un análisis de cada tipo de ataque que se presenta en la web, lo que lleva a cabo la generación de categorías estándar para el análisis de cada ataque y así deducir cuales son los ataques más riesgosos para las organizaciones. Estas categorías fueron las que se escogieron como las clases del modelo ontológico.

Clases del modelo ontológico

Agentes de amenazas o *threat agents*. El Agente de amenaza es un término que se utiliza para identificar un individuo o grupo que puede manifestarse como una amenaza. Es fundamental identificar quién querría explotar los activos de una empresa, y cómo podrían usarlas contra la empresa. Es así como (Hancock, 1998) definió el modelo para los agentes de amenazas como se define en la siguiente ecuación:

$$Threatagents = Capabilities + Intentions + PastActivities$$

Ecuación 2. Agentes de amenaza, OWASP *Threat Agent* (OWASP, 2011)

Este modelo matemático, es una actividad para entender la seguridad en una aplicación; midiendo la vulnerabilidad, las contramedidas relacionadas y el impacto, no se requiere una amenaza, porque la amenaza existe incluso si el objetivo está bien protegido contra ella.

Vectores de ataque o *attacks vectors*. Un vector de ataque es una ruta o un medio por el que un *hacker* puede tener acceso a un ordenador o servidor de red, con el fin de entregar una carga útil o resultado malicioso. El ataque permite a los *hackers* explotar las vulnerabilidades del sistema, incluyendo el elemento humano.

Debilidades de seguridad o *security weakness*. Una vulnerabilidad en cualquier sistema se define como un fallo, laguna, debilidad o defecto existente en el sistema, que puede ser explotada por un usuario no autorizado con el fin de realizar algún ataque o para obtener acceso a

ISSN 2145–549X | ISSN 2422-4200, Vol. 8, No. 1, Julio - Diciembre 2016

los datos almacenados (Mukesh Kumar Gupta, 2014).

Impacto técnico o *technical impact*. El impacto técnico de los ataques web puede permitir el ataque a varias cuentas de portales en internet, consiguiendo cuentas preferenciales de las distintas bases de datos. Así, los atacantes pueden ejecutar *scripts* en el navegador de la víctima para secuestrar sesiones de usuario, desconfigurar sitios web, insertar contenido hostil, redirigir a los usuarios, secuestrar el navegador usando el *malware* del usuario, la falta de rendición de cuentas, o la denegación de acceso (OWASP top 10, 2016).

Impacto de negocios o *business impacts*. El impacto de negocio es prácticamente la razón de la creación del *honeypot* y la mayoría de aplicaciones web actuales; garantizar la seguridad de transacciones e información sensible para las personas o empresas, se convierte en un tema de primera prioridad. A partir de la correcta implementación de parámetros y técnicas de seguridad, se garantizará un nivel de seguridad alto en cuanto a transacciones monetarias e información crediticia, brindando la oportunidad a compañías y personas de usar la web para su comodidad financiera y personal. Kumar, Gupta (2014) define un alto impacto en los usuarios pues estos no podrían desarrollar sus tareas de forma normal; en el caso de un sistema de producción sería un error crítico pues retrasaría toda la operación. La figura 2 presenta la vista que da *protégé* en la generación del modelo ontológico en vista de árbol.

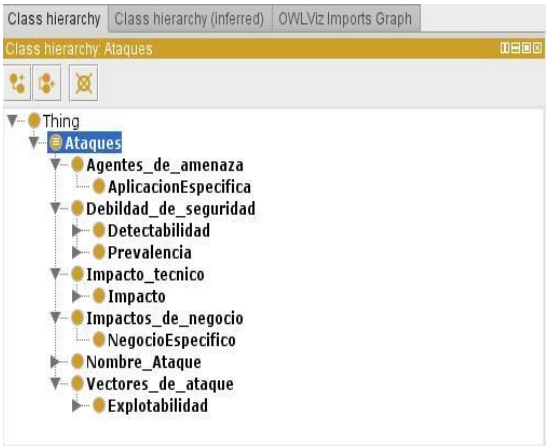

Figura 2. Vista en forma de árbol de las clases del modelo ontológico. Fuente: autores.

Propiedades del modelo ontológico

A continuación, una vez planteado el modelo, se definen las propiedades del modelo ontológico. Las propiedades OWL representan relaciones entre dos objetos o individuos o propiedades de una clase. Para el modelo planteado, se generó una propiedad objeto llamada ***Tiene riesgos*** el cual se deriva de la misma cantidad de clases creadas con el mismo nombre, solo que anteponiendo el prefijo ***Tiene***. A esas propiedades se crean propiedades inversas que son la misma cantidad de clases, pero con un prefijo ***Es*** o ***Son*** y el sufijo ***de***. Dependiendo de la clase a que pertenezcan se pueden aplicar ciertas propiedades (transitiva, funcionales entre otras).

Lo siguiente es especificar el dominio y rango de las propiedades dadas a las primeras que generamos; se aplica que el dominio es ***Ataques*** y su rango es cada clase relacionada, pero para las propiedades inversas el dominio y el rango se invierten, al ser propiedades inversas.

Lo que sigue para generar el modelo ontológico, es generar los constructores de clases y axiomas para las clases, propiedades e individuos. En los constructores se encuentran una gran variedad de conectores lógicos como: cuantificadores, equivalencias, nominales entre otras. La figura 3 muestra las diferentes características y propiedades que muestra *protégé* para una clase entre los usos y las descripciones.

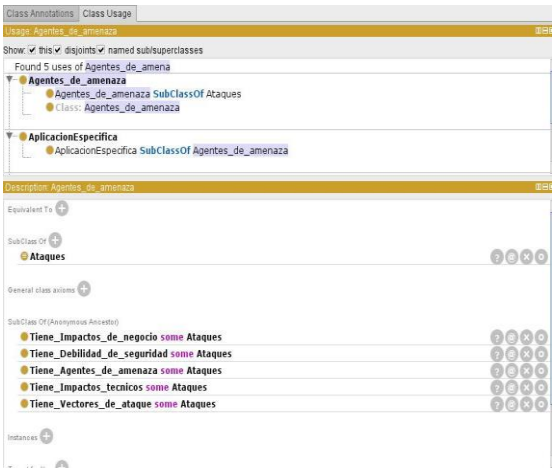

Figura 3. Características de una clase. Fuente: autores.

Protégé genera una serie de vistas que muestran las diferentes características que se presentan para una subclase tales como los usos, descripciones, equivalencias, subclases, clases generales, instancias, entre otras.

Una parte importante de nuestro modelo ontológico, es generar los ataques escogidos anteriormente y asignarles las propiedades y constructores faltantes. En la figura 4 se presentan las construcciones que se generan para una clase específicamente y para uno de los ataques: en este caso *broken authentication*.

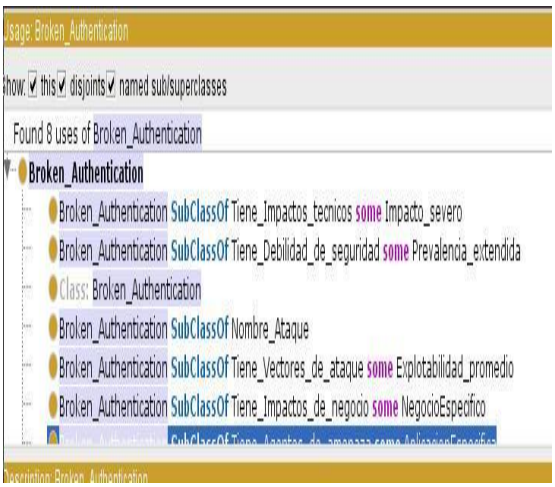

Figura 4. Subclases y las relaciones de cada clase. Fuente: autores.

En la figura 4 se muestra cómo se realizan las construcciones finales para un ataque escogido y cómo quedan las propiedades; se puede observar que tiene las mismas clases que se encuentran en la página de la OWASP para el ataque escogido. Lo último que se realiza, es la creación de las instancias que serán objetos concretos de ataques, en este caso los mismos ataques que se incluyeron en la clase ***Nombre Ataque***. Esto último se presenta en la figura 5.

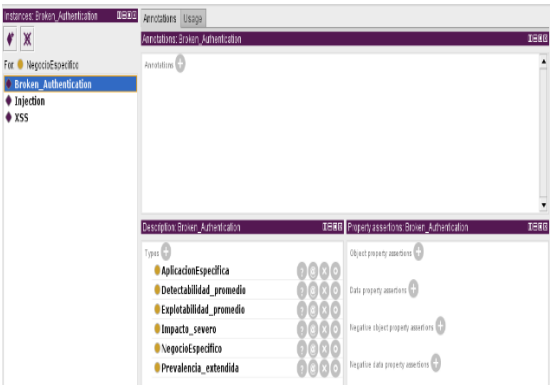

Figura 5. Presentación de las instancias de tipo Ataque. Fuente: autores.

La parte final en la realización del modelo ontológico, es utilizar un razonador el cual cumple la función de un compilador de cualquier otro lenguaje de programación. Este razonador lo hace pero de manera semántica. Entre los razonadores utilizados se encuentran ***HermiT*** versión ***1.3.8***, ***E.L.K*** versión ***0.4.3, FaCT++, Ontop*** versión ***1.17.1, Pellet*** y ***jcel.***, razonadores de ontologías escritas utilizando el Lenguaje de Ontologías *web* (OWL). Dado un archivo OWL, estos razonadores pueden determinar si es consistente la ontología, identificar relaciones subsunción entre clases, instancias y más relaciones semánticas. Un ejemplo del uso de estos razonadores se presenta en la figura 6 mediante la ejecución del razonador ***HermiT.***

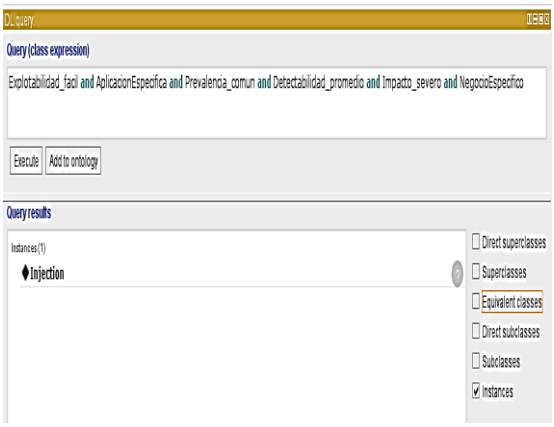

Figura 6. Aplicación del razonador *HermiT* para propiedades de un ataque de tipo *Injection*. Fuente: autores.

El análisis del razonador aplicado en el modelo ontológico, muestra que no se encontraron inconsistencias entre los axiomas, clases, datos, objetos y relaciones del modelo. Esto nos muestra cómo *protégé* nos facilita la comprobación del modelo ontológico, gracias a los diferentes razonadores que posee, sin importar qué tan grandes sean los modelos que sean generados. Estos razonadores son parte primordial en el desarrollo de la *web* semántica ya que permite mostrar las inconsistencias generadas en modelos complejos.

Para corroborar todos los elementos del modelo ontológico *protégé* genera métricas sobre el modelo realizado mostrando axiomas, clases, datos, objetos relacionales, que en el caso de estudio generado, se muestran en la figura 7. Estas métricas permiten ejercer un mejor control de los modelos que, entre más complejos sean, mayores beneficios se obtienen de estas métricas.

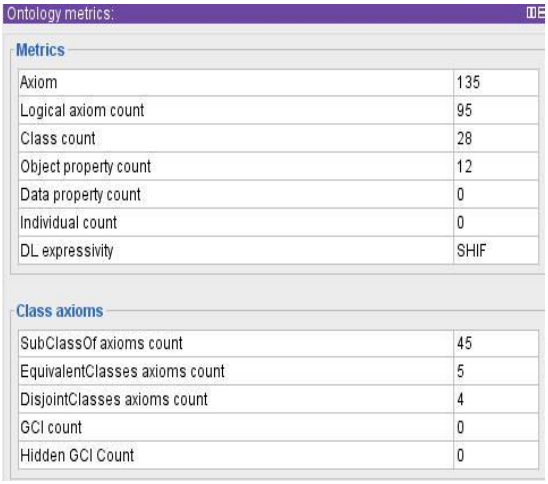

Figura 7. Tabla de métricas del modelo Ontológico presentada por *protégé*.

Visualización del modelo ontológico generado

En *protégé*, existen *plugins* o herramientas que nos permiten ver la visualización gráfica de los modelos ontológicos creados; entre ellas *OntoGraf y OWLViz,* ver figuras 8 y 9 respectivamente.

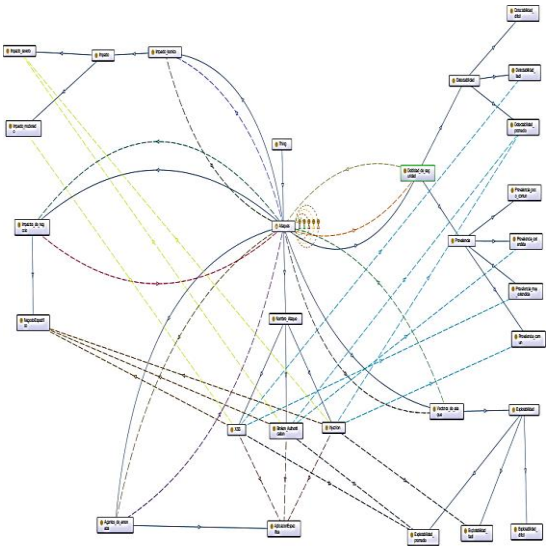

Figura 8. Modelo ontológico generado por el *plugin OntoGraf* de *Protégé*

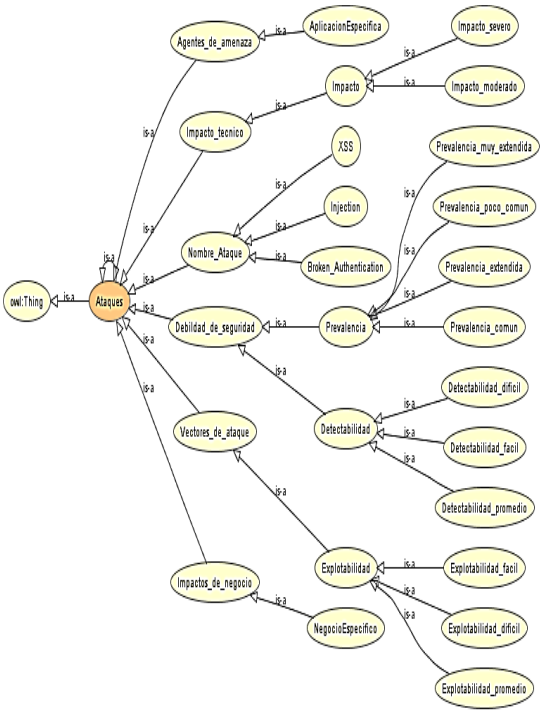

Figura 9. Visualización del modelo ontológico planteado, por el *plugin* OWL*Viz*

Conclusiones

Los modelos ontológicos organizan la información para limitar la complejidad de diferentes temas. Esto permite un mayor entendimiento de temas con una complejidad importante, al realizar relaciones semánticas entre términos claves de los distintos temas. Es así como el modelo ontológico planteado muestra las conexiones semánticas presentadas para los tres ataques más importantes en seguridad informática especialmente aplicaciones *web*.

Utilizar razonadores ontológicos permite determinar ataques de seguridad *web*, por sus propiedades tecnológicas para la publicación de datos legibles por parte de aplicaciones informáticas, pero se deben tener en cuenta las conexiones y propiedades de las clases e instancias que se generan, y realizar pruebas con distintos razonadores.

La OWASP genera el informe del top 10 de vulnerabilidades cada tres (3) años, por lo que la actualización de la información debe llegar en el año 2016. En el corto plazo, se debe actualizar el modelo ontológico por si se genera algún cambio en el *top*, para mantener el modelo acorde a los nuevos desafíos que en seguridad de la información.

El proyecto *Honeynet* mantiene proyectos en seguridad informática, que se deben revisar para identificar los proyectos más actualizados para aprender cómo identificar ataques y virtualizar en el modelo ontológico.

Como trabajo futuro, queda la implementación del modelo dentro de un escenario real de comunicaciones, para el planteamiento de un *framework* que permita llevar a cabo la visualización de ataques más frecuentes, para toma de decisiones a partir de vulnerabilidades más frecuentes en una red. De manera complementaria, se pretende medir escenarios de trabajo con el propósito de determinar niveles de confianza dentro de entornos de trabajo colaborativo, tal y como se platea por autores como Vásquez & López (2015), así como llevar a cabo escenarios de aplicación mediante escenarios virtualizados Daas (Rodríguez et ál., 2016).

Referencias bibliográficas

Akrivi, K., Elena, T., Constantin, H., Georgios, L. & Costas, V. (2006). *A comparative study of four ontology visualization techniques in protégé : Experiment setup and preliminary results*. In Tenth International Conference on Information Visualisation (IV'06) (pp. 417-423). IEEE.

Carreño Bustamante, M. (2014). La formación de los estudiantes de derecho, bajo el paradigma de la investigación sociojurídica. *Revista Logos Ciencia & Tecnología,* 5(2), 289-297. doi:http://dx.doi.org/10.22335/rlct.v5i2.113

Chang, J. C. H. & Tsai, Y. L. (2010). *Design of virtual honeynet collaboration system in existing security research networks. In Communications and Information Technologies* (ISCIT), 2010 International Symposium on (pp. 798-803). IEEE.

ISSN 2145–549X | ISSN 2422-4200, Vol. 8, No. 1, Julio - Diciembre 2016

Djanali, S., Arunanto, F. X., Pratomo, B. A., Baihaqi, A., Studiawan, H. & Shiddiqi, A. M. (2014). *Aggressive web application honeypot for exposing attacker's identity. In Information Technology, Computer and Electrical Engineering* (ICITACEE), 2014 1st International Conference on (pp. 212-216). IEEE.

Gallego, E. & de Vergara, J. E. L. (2004). Honeynets: aprendiendo del atacante. In IX Congreso Nacional de Internet, Telecomunicaciones y Movilidad.

Gupta, M. K., Govil, M. C. & Singh, G. (2015). *Predicting Cross-Site Scripting (XSS) security vulnerabilities in web applications. In Computer Science and Software Engineering* (JCSSE), 2015 12th International Joint Conference on (pp. 162-167). IEEE.

Gupta, M. K., Govil, M. C. & Singh, G. (2014). *Static analysis approaches to detect SQL injection and cross site scripting vulnerabilities in web applications: A survey. In Recent Advances and Innovations in Engineering* (ICRAIE), 2014 (pp. 1-5). IEEE.

Karthik, S., Samudrala, B. & Yang, A. T. (2009). Design of Network Security Projects Using *Honeypot*s. *Journal of Computing Sciences in Colleges,* 20(4).

Kwon, D., Hong, J. W. K. & Ju, H. (2012). *DDoS attack forecasting system architecture using honeynet. In Network Operations and Management Symposium* (APNOMS), 2012 14th Asia-Pacific (pp. 1-4). IEEE.

Hancock, B. (1998). Steps to a successful creation of a corporate threat management plan. *Computer Fraud & Security,* 1998(7), 16-18.

Horridge, M., Knublauch, H., Rector, A., Stevens, R. & Wroe, C. (2004). A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0. University of Manchester.

Huluka, D. & Popov, O. (2012). Root cause analysis of session management and broken authentication vulnerabilities. In Internet Security (WorldCIS), 2012 World Congress on (pp. 82-86). IEEE.

Ma, J., Chai, K., Xiao, Y., Lan, T. & Huang, W. (2011). High-Interaction *Honeypot* System for SQL Injection Analysis. In Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on, 3, pp. 274-277). IEEE.

Memari, N., Hashim, S. J. B. & Samsudin, K. B. (2014). Towards virtual *honeynet* based on LXC virtualization. In Region 10 Symposium, 2014 IEEE (pp. 496-501). IEEE.

Noy, N. F., Crubézy, M., Fergerson, R. W., Knublauch, H., Tu, S. W., Vendetti, J. & Musen, M. A. (2003). Protégé -2000: an open-source ontology-development and knowledge-acquisition environment. In AMIA Annu Symp Proc . 953, 953).

OWASP, T. (2016). *Top* 10. The Ten Most Critical Web Application Security Risks. https://www.owasp.org/index.php/Main_Page Last access (21 July 2016)

Papapanagiotou, K. (2013). *OWASP Hackademic*: a practical environment for teaching application security. In AppSec USA 2013. Owasp.

Rodríguez, J. A. F., Marín, C. E. M., Bonilla, J. A. R. & García, P. A. G. (2016). Hacia la virtualización de escritorios para la entrega de ambientes académicos basados en DaaS. Revista *Logos Ciencia & Tecnología*, 7(2), 114-124.

Sadeghian, A., Zamani, M. & Ibrahim, S. (2013). SQL injection is still alive: a study on SQL injection signature evasion techniques. In Informatics and Creative Multimedia (ICICM), 2013 International Conference on (pp. 265-268). IEEE.

Sqalli, M. H., Firdous, S. N., Baig, Z. & Azzedin, F. (2011). An Entropy and Volume-Based Approach for Identifying Malicious Activities in *Honeynet* Traffic. In Cyberworlds (CW), 2011 International Conference on (pp. 23-30). IEEE.

Uschold, M. & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(02), 93-136.

Vásquez, L. M. L. & López, M. D. R. (2015). Implementación de una herramienta virtual para la

determinación de la confianza. Revista *Logos Ciencia & Tecnología*, 6(2), 177-187.

Vergel, M., Martínez, J. & Zafra, S. (2016). Factores asociados al bullying en instituciones de educación superior. *Revista Criminalidad,* 58 (2): 197-208. http://www.policia.gov.co/imagenes_ponal/dijin/revista_criminalidad/v58n2/v58n2a11.pdf

Vergel Ortega, M., Duarte, H., & Martínez Lozano, J. (2016). Desarrollo del pensamiento matemático en estudiantes de cálculo integral su relación con la planificación docente. Revista Científica, 3(23), 17-29. doi:http://dx.doi.org/10.14483/udistrital.jour.RC.2015.23.a2

Watson, D. & Riden, J. (2008). *The honeynet project: Data collection tools, infrastructure, archives and analysis*. In WOMBAT Workshop on Information Security Threats Data Collection and Sharing (pp. 24-30). IEEE.

Wazzan, M. A. & Awadh, M. H. (2015). Towards Improving Web Attack Detection: Highlighting the Significant Factors. In IT Convergence and Security (ICITCS), 2015 5th International Conference on (pp. 1-5). IEEE.

Yao, Y., Lv, J. W., Gao, F. X., Yu, G. & Deng, Q. X. (2009). Detecting and Defending against Worm Attacks Using Bot-*honeynet*. In Electronic Commerce and Security, 2009. ISECS'09. Second International Symposium on (Vol. 1, pp. 260-264). IEEE.

Yu, Y. T. & Hsu, C. C. (2011). A structured ontology construction by using data clustering and pattern tree mining. In Machine Learning and Cybernetics (ICMLC), 2011 International Conference on (Vol. 1, pp. 45-50). IEEE.

Zurutuza, U., Ezpeleta, E., Herrero, Á. & Corchado, E. (2011). Visualization of misuse-based intrusion detection: Application to *honeynet* data. In Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011 (pp. 561-570). Springer Berlin Heidelberg.