2015

Universitat Oberta de Catalunya

Juan Clavero

Dirige: Cristina Pérez Solà

Universitat Autònoma de Barcelona

Universitat Oberta de Catalunya

[DETECCIÓN DE INTRUSOS CON SNORT]

PROYECTO DEL POSTGRADO EN SEGURIDAD EN REDES Y SISTEMAS

Sinopsis

Una información importante para proteger cualquier red de accesos no permitidos es el conocimiento del tráfico que puede circular por dicha red, discriminando el tráfico deseado y permitido del no deseado. Una herramienta como snort permite, en base a una reglas predefinidas, catalogar el tráfico existente y realizar una serie de acciones frente al indeseado. Este proyecto simula de forma simplificada una red privada en la que se desea conocer si se realizan una serie de intentos de ataque desde el exterior.

Abstract

Every network administrator needs to know the traffic of its network in order to protect it from forbidden access, distinguishing wanted and allowed traffic from unwanted one. A tool like snort allows to classify the existing traffic and to perform a series of actions to tackle the undesirable. This project simulates a simplified private network in which there's a need to know if some kind of attacks from the outside is performed.

Tabla de contenido

1 Introducción

1.1 Problema a resolver

Se plantea la protección de un entorno doméstico semi-profesional: un profesional autónomo, trabajando desde el domicilio, ofrece un servicio web a sus clientes para la gestión de incidencias basado en tickets para facilitar la gestión de los proyectos contratados.

Este servicio web se basa en la aplicación OTRS, implementado en Perl y que se puede ejecutar en la pila LAMP. A pesar de ser una aplicación bastante robusta y con años de historia, el profesional no se fía y decide hacer todo lo que pueda para evitar ataques de denegación de servicio (DoS), que impidan a sus usuarios acceder al sistema y ponerse en contacto con él, y ataques de inyección de instrucciones SQL, que permiten a los atacantes obtener información privada.

En esta situación, se propone dirigir todo el tráfico entrante en su domicilio a un equipo bastión, que realizará las tareas de firewall y de análisis de tráfico entrante antes de redirigirlo hacia el equipo servidor o hacia el punto de acceso a la red doméstica.

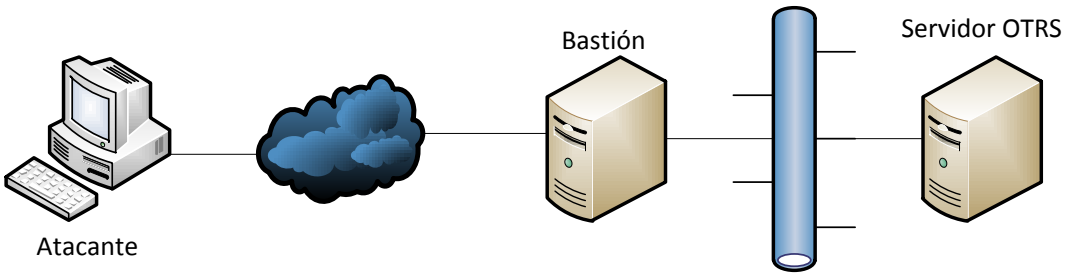


Ilustración 1. Esquema del entorno

1.2 Objetivos

Los objetivos del proyecto son:

1. Evaluar las necesidades de seguridad del escenario planteado.
2. Estudiar las alternativas disponibles en el mercado para conseguir la seguridad esperada en el escenario planteado, tanto por su capacidad de protección como por los requisitos hardware para su ejecución y su coste económico.
3. Instalar y configurar snort en la máquina bastión.
4. Diseñar un plan de pruebas adecuado en el que evaluar las capacidades reales del aplicativo seleccionado, tanto en cuanto a su capacidad para detectar ataques reales como para evitar la generación de falsas alertas.
5. Ejecutar el plan de pruebas y comprobar el desempeño del aplicativo.

1.3 Metodología

Debido a las capacidades actuales de simulación, la misma máquina virtual realizará las tareas de bastión y de servidor web. Esta funcionará con una versión de Linux basada en Ubuntu 14.04, en la que se configurará iptables (como firewall), snort (como detector de intrusiones),

apache (como servidor web), MySQL (como servidor de bases de datos) y OTRS (como aplicativo web).

Esta máquina virtual será atacada desde otra máquina en la que se instalará la distribución Kali Linux para realizar ataques de escaneo de vulnerabilidades web y se lanzarán solicitudes simultáneas para activar la alerta de intento de DoS. Tras estos ataques se comprobará el registro de snort para verificar su registro. También se realizará un uso habitual del aplicativo web para comprobar si el sistema detector de intrusiones lanza falsos positivos.

1.4 Planificación

1.4.1 Tareas

1. Análisis Estado del Arte

2. Configuración Bastión
   - 2.1. Creación Máquina Virtual
   - 2.2. Instalación Sistema Operativo
   - 2.3. Configuración iptables
   - 2.4. Comprobación iptables
   - 2.5. Instalación Snort
   - 2.6. Configuración Snort
   - 2.7. Instalación Apache
   - 2.8. Instalación MySQL
   - 2.9. Instalación OTRS
   - 2.10. Configuración OTRS

3. Configuración Atacante
   - 3.1. Creación Máquina Virtual
   - 3.2. Instalación Kali Linux

4. Fase de Ataque
   - 4.1. Uso normal
   - 4.2. Ataques vulnerabilidades
   - 4.3. Ataques DOS

5. Fase de Análisis
   - 5.1. Recogida de datos
   - 5.2. Análisis de Datos

1.4.2 Planificación temporal

Creación y configuración de la máquina Bastión: 15 días, de 20/03/15 a 05/04/2015

Configuración de la máquina Atacante: 7 días, de 06/04/2015 a 12/04/2015

Fase de Ataque: 10 días, de 20/04/2015 a 29/04/2015

Fase de Análisis: 15 días, de 08/05/2015 a 23/05/2015

En el Anexo Diagrama de Gantt se presenta un desglose pormenorizado de la planificación temporal de cada tarea, con distintas versiones según ha ido progresando el proyecto.

1.5 Estado del arte

Los sistemas de detección de intrusos (IDS) pretenden mejorar la seguridad de un sistema avisando a los administradores cuando se detecten comportamientos relacionados con ataques conocidos o que se alejen del comportamiento habitual de sus usuarios o equipos. Para ello,

monitorizan el tráfico que circula por la red y por algunos sistemas y actúan frente a patrones de tráfico conocidos, lo que permite a los administradores conocer claramente qué está pasando por la red, almacenar información sobre las actividades observadas y recibir alertas ante patrones específicos. Debido a su naturaleza detectora, los IDS no pueden prevenir ataques ni ofrecer protección sobre los recursos internos, sólo constituyen un nivel adicional de seguridad. Dependiendo de la ubicación desde la que se obtienen los datos, se pueden clasificar en IDS de red (*Network IDS*) que capturan la información en la interfaz de red, o IDS de equipo (*Host IDS*), que analizan información de eventos del sistema operativo de la máquina. Cuando un IDS tiene capacidad para modificar la configuración de elementos de red, lo que le permite una primera reacción ante ataques, se conoce como Sistema de Prevención de Intrusos (IPS). También se pueden clasificar según su funcionamiento, ya sea por detección de patrones conocidos o por detección de anomalías, pero la problemática de configuración de estos últimos, que necesitan conocer los perfiles de los usuarios legítimos, hace que la mayoría de los IDS disponibles sea de detección de patrones.

Entre las alternativas de software libre, los NIDS Snort, Suricata y Kismet y los HIDS OSSEC y Samhain. Las principales empresas del sector de las telecomunicaciones y de la seguridad informática también ofrecen sus productos IDS/IPS, como por ejemplo la serie IPS 4200 de Cisco, Proventia IPS de IBM y McAfee *Network Security Platform*, todos ellos NIDS.

Snort es el estándar de facto entre los IDS de red. Creado en 1998, tiene una gran comunidad que ofrece soporte y herramientas libres para facilitar su configuración y gestión. Suricata, por su parte, ofrece compatibilidad con las reglas creadas para Snort y añade, entre otras, la capacidad de proceso multihilo, incluso usando las GPU de las tarjetas gráficas. Kismet constituye la referencia de los IDS para redes inalámbricas. Se encarga menos de los paquetes individuales que del funcionamiento de los protocolos inalámbricos (WiFi, 802.11), siendo capaz de encontrar, por ejemplo, puntos de acceso no autorizados o puntos de acceso cercanos con el mismo nombre que podrían servir para iniciar un ataque de intermediario (*Men in the Middle*).

Pasando a los HIDS, OSSEC tiene una arquitectura de cliente-servidor: los clientes se instalan en las máquinas que dan servicio y transfieren los logs de sistema y de aplicativo al servidor para su procesamiento y análisis. Samhain usa una arquitectura muy similar pero son los clientes instalados en las máquinas servidoras las que procesan los logs antes de enviar la información al servidor central: esto limita el ancho de banda necesario y la carga del servidor Samhain a costa de aumentar la carga en los servidores corporativos.

Detallando un poco las propuestas comerciales, la serie IPS 4200 de Cisco dispone de reconocimiento de patrones y de "análisis de reputación" global que puede enviar actualizaciones a los clientes en cuestión de minutos, además de integrarse fácilmente y permitir su gestión a través de los sistemas de gestión de la compañía. Proventia IPS está diseñado con una arquitectura modular que facilita la escalabilidad para facilitar la adaptación y maximizar la capacidad de detección y ofrece como funcionalidad estrella el módulo de análisis de protocolo PAM que promete capacidad de inspección completa de los paquetes. McAfee NSP es la única solución IPS que cuenta con la certificación Multi-Gigabit IPS del grupo NSS y ofrece un sistema centralizado de gestión de políticas y de hasta 1000 nodos IPS.

2 Conceptos previos

2.1 Sistemas de Detección de Intrusos

Como se ha explicado anteriormente, un sistema de detección de intrusos es un sistema que monitoriza el estado de los sistemas pertenecientes a una red para determinar si se produce un acceso no autorizado.

En el presente caso, Snort es un sistema de detección de intrusos de red, ya que monitoriza los paquetes de datos recibidos por una interfaz de la máquina en la que se encuentra instalado para compararlos frente a patrones de comportamientos conocidos y actuar frente a posibles intrusos o peligros. De esta manera, permite al administrador de la red recibir notificaciones en caso de que se produzcan en la red tráfico considerado perjudicial para la seguridad de las aplicaciones y de la información existentes en la red. Para ello, se basa en la definición de reglas de detección: líneas de texto que describen las características presentes en el tráfico analizado para que la regla se considere como cumplida y la acción que debe realizar snort ante el tráfico que cumpla la regla.

2.2 Ataques de Inyección de SQL

Los ataques de inyección SQL confían en la dejadez de los programadores a la hora de traducir las solicitudes de los usuarios de las aplicaciones en sentencias SQL para extraer información de la Base de Datos.

Una parte importante de muchas aplicaciones web es mostrar al usuario la información, parcial o total, almacenada en la Base de Datos. Para ello, ofrece un URL del que una parte se considera variable y se entiende como el identificador del objeto que se desea ver. Por ejemplo, una tienda de zapatos podría ofrecer el URL `…/zapatos?id=5` para obtener información sobre el par de zapatos almacenado con el identificador 5. El inconveniente es que, para obtener información, muchos programadores simplemente trasladaban al servidor de Bases de Datos el contenido del URL tras el componente conocido '`id=`':

```
select * from zapatos where id = 5;
```

Ilustración 2. Sentencia SQL vulnerable

Aprovechándose de esta situación, un ataque de inyección SQL modifica el valor tras el componen conocido para colocar un texto que permita construir una nueva sentencia SQL con resultados no previstos por el programador de la herramienta. Así, siguiendo con el caso anterior, el uso del URL `…/zapatos?id=5; select * from usuarios` produciría una sentencia SQL válida que facilitaría, además de información sobre el par de zapatos con identificador 5, el contenido de la tabla usuarios, si existiera en la Base de Datos:

```
select * from zapatos where id = 5; select * from usuarios;
```

Ilustración 3. Sentencia SQL tras una inyección de código

El peligro no es únicamente la visualización de información restringida directamente hacia el navegador: el sólo hecho de poder forzar un error en el servidor mediante la inclusión de código SQL puede producir la filtración de información. Así, el URL `…/zapatos?id=5 and 1=0` provocaría una sentencia SQL válida que no devolvería información, lo cual podría llevar a programar

un ataque en el que se fuera extrayendo, letra a letra, el contenido de la tabla usuarios. Este tipo de ataque se llama ataque de inyección de SQL a ciegas.

`…/zapatos?id=5 and exists ( select * from usuarios )`	Devolverá información si existe la tabla usuarios
`…/zapatos?id=5 and 10 = select count(*) from usuarios`	Devolverá información si el número de usuarios es 10.
`…/zapatos?id=5 and 65 = select ASCII( select user from usuarios limit 1 )`	Indicaría si la primera letra del primer usuario es una 'A' mayúscula (65 en código ASCII) si la aplicación usa MySQL.

Ilustración 4. Varias URL con SQL para un ataque a ciegas

Cualquier medio por el que se proporcione un parámetro a un sitio web para su uso en una Base de Datos es susceptible de sufrir un ataque de inyección SQL:

- parámetros GET (en el URL de la solicitud, como en los ejemplos vistos hasta ahora);
- parámetros POST (los parámetros van en el cuerpo del mensaje de la solicitud, no en el URL);
- cookies: El protocolo HTTP no tiene concepto de sesión y cada petición es independiente de las anteriores y las posteriores. Para proporcionar un entorno a las solicitudes realizadas a un servidor, éste puede proporcionar pequeños segmentos de información al navegador para que sean reenviados en cada solicitud y pueda reconstruirse la sesión de navegación del usuario. Estos segmentos se conocen como cookies y uno de los primeros pasos para realizar la reconstrucción es comprobar si existe en la Base de Datos una sesión similar al contenido de la cookie recibida, con lo que puede suponer un vector de inyección de código SQL;
- cabeceras HTTP: en ocasiones, los servicios utilizan parte de la cabecera HTTP para afinar el comportamiento (como puede ser el campo '*Referer*', que indica la web desde la que se llegó al sitio web actual). Si se emplea el contenido de esta cabecera en una consulta de Base de Datos también constituirá un punto vulnerable de ataque por inyección de código SQL.

2.3 Ataques de Denegación de Servicio

Un ataque DoS pretende anular la capacidad de comunicaciones de la máquina objetivo saturando sus canales de comunicación. Para ello, envía una gran cantidad de solicitudes sin esperar a recibir respuesta desde la víctima. Si la capacidad de los atacantes es superior a la de la víctima, la capacidad de ésta se irá reduciendo hasta que quedará de forma efectiva desconectada de la red.

Los ataques de este tipo más efectivo son los distribuidos (DDoS), en los que muchas máquinas se sincronizan para atacar a unas pocas máquinas víctimas. Cada vez en más ocasiones, las máquinas participantes en un ataque DDoS son zombies que han sido infectados y controlados por redes botnet. Las capacidades de simulación impiden enfocarnos en este caso.

3 Escenario planteado

El escenario planteado persigue proteger el servicio de gestión de incidencias ofrecido por un profesional a los clientes de sus servicios.

OTRS es una herramienta web de código libre para la gestión de incidencias. Se basa en el uso de tickets, documentos que contienen toda la información relevante a un incidente, como puede ser su prioridad, el usuario que la ha comunicado, a qué servicio afecta, etc. Desde el punto de vista del cliente, se puede comprobar el estado y la evolución de los incidentes reportados (los tickets), mientras que desde el punto de vista del profesional permite organizar los tickets en áreas (colas), priorizarlos, anotar acciones y documentarlos. Con el paso del tiempo, el uso de la herramienta ofrece un histórico de soluciones aplicadas que facilitan la resolución de nuevos incidentes. OTRS está desarrollada en perl por la empresa alemana OTRS AG, que también ofrece personalizaciones y soporte corporativos, y está disponible de forma gratuita en la plataforma github[1]. Se ha elegido OTRS por ser una aplicación de código libre con una gran comunidad de soporte detrás y actualizaciones de seguridad en cuanto se detectan fallos en la misma.

Al ser una herramienta web, requiere un servidor web que reciba las solicitudes http y las traslade (y traduzca) al idioma en el que la aplicación OTRS está escrita. Para ello se emplea el servidor web Apache con el paquete mod_perl, que permite la ejecución de scripts.

Además del servidor web, OTRS necesita un servicio para almacenar la información de gestión de la herramienta: un servidor de Bases de Datos en el que se almacenará tanto la información facilitada por los clientes como la añadida por los gestores del ticket y la de la propia herramienta para permitir su funcionamiento. Puede emplear tanto Bases de Datos relacionales como NoSQL, pero por su facilidad de instalación se ha optado por MySQL. La interfaz de comunicación entre OTRS y MySQL es TCP/IP sobre el puerto 3306 de la máquina, si bien está configurado para recibir conexiones sólo desde la propia máquina y se considera lejos del alcance del presente proyecto el comprobar que no se produzcan accesos no deseados directamente al servidor de Bases de Datos ya que normalmente estaría en una máquina detrás del servidor web y no accesible directamente desde el exterior.

En cuanto a hardware, el escenario consta de 3 máquinas:

1. Servidor de aplicaciones: equipo situado en la zona perimetral del entorno doméstico, accesible tanto desde el interior de la red como desde el exterior y alberga el software necesario para ofrecer el servicio OTRS;
2. Bastión: equipo situado en la unión entre la zona perimetral del entorno doméstico y el exterior destinado a proteger el entorno doméstico, alberga el sistema de detección de intrusos; y
3. Atacante: máquina externa que intenta disturbar el servicio ofrecido por el servidor de aplicaciones.

[1] https://github.com/OTRS/otrs

Si bien lógicamente sean máquinas distintas, la capacidad de simulación con la que se dispone obliga a unir servidor de aplicaciones con el bastión, alojando en una máquina todos los servicios necesarios para la ejecución de OTRS y el sistema de detección de intrusos para proteger el entorno. En una situación real, esta decisión sería inaceptable por exponer los servicios alojados directamente al exterior.

Igualmente, por simplicidad de la simulación, tanto atacante como bastión están conectados a la misma red local.

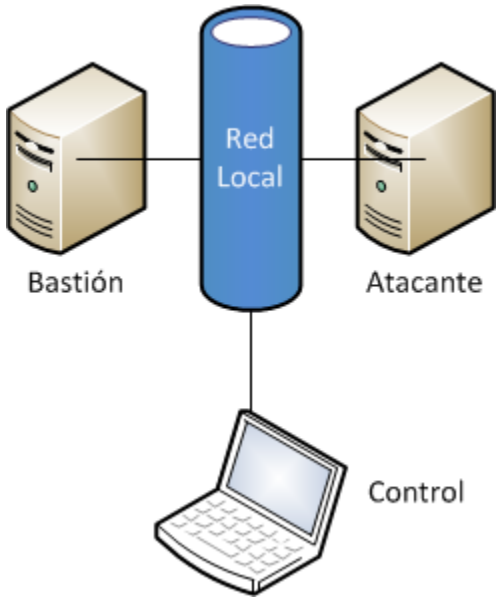


Ilustración 5. Diagrama de la simulación

3.1 Bastión

El bastión es la máquina que deberá recibir las peticiones de los clientes y dirigirlas al aplicativo OTRS para que realice su función, pero también deberá asegurarse de que no hay accesos no autorizados que impidan el acceso a los servicios ni que puedan obtener información de éstos.

3.1.1 Configuración

Se crea una máquina virtual con capacidad suficiente para ejecutar los servicios necesarios para la aplicación OTRS (esto es, apache2 con mod_perl y MySQL). En ella, se instala la distribución debian para servidores de 64 bits (concretamente, debian 7.8.0 amd64). Como esta distribución trae activado iptables, se añaden las reglas para bloquear todo el tráfico entrante por la interfaz de red conectada al exterior excepto las solicitudes de conexión al puerto https y las conexiones ya establecidas. Por limitaciones de la simulación, se permite también la conexión por SSH para el control remoto por la misma interfaz de red.

Una vez protegida la máquina, se instalan, con el comando apt-get, los servicios apache2 y MySQL y la aplicación OTRS. El comando apt-get se encarga de la instalación de todos los requisitos de estos programas. La única configuración necesaria en esta fase ha sido la definición de la contraseña de la cuenta del usuario privilegiado *root* en MySQL y la información de inicio de sesión privilegiado en MySQL para que OTRS creara la estructura de Base de Datos y la rellenara con el contenido mínimo necesario para el funcionamiento de la herramienta(cuenta de administración de la aplicación, niveles de prioridad de los tickets, colas para la gestión de los tickets, etc.). Después, se ha configurado apache2 para activar el acceso por https (con un certificado autofirmado) y desactivar el acceso por http.

3.1.2 Snort

Se ha dejado para el final la instalación y configuración de snort, también realizada con apt-get. La configuración inicial solicita detalles mínimos sobre la configuración de la red local (rango de direcciones IP). Las reglas incluidas en el paquete son de abril 2014, así que se decide

instalar también la aplicación oinkmaster para actualizarlas. Por desgracia, oinkmaster no está actualizado (última versión es de 2006) y presenta problemas a la hora de descargar las nuevas reglas, con lo que se ha optado por realizar la descarga de un paquete de reglas actualizado e instalarlo. Aun así, un análisis de las reglas instaladas muestra que ninguna de ellas sería de utilidad en el ejercicio propuesto, así que se decide eliminar las reglas recientemente descargadas y escribir un set de reglas personalizado.

3.1.3 Reglas

Como se ha comentado con anterioridad, snort se basa en el uso de reglas para establecer patrones de tráfico no deseados y la acción que se debe realizar ante éstos. Para ello, las reglas se componen, por un lado, de **cabecera**, donde se incluye la acción asociada a la regla, el tipo de paquete (ICMP, TCP, UDP…), direcciones de origen y de destino del paquete, etc., y del campo **opción**, donde se pueden añadir filtros adicionales sobre el contenido del paquete analizado para determinar que se active la acción asociada a la regla.

```
<Acción> <Protocolo>
<IP_Origen> <Puerto_Origen> -> <IP_Destino> <Puerto_Destino>
{Opción_1; …; Opción_N}
```

Ilustración 6. Prototipo de regla de snort

Algunas de las posibles acciones de snort para los paquetes que cumplen una regla son registrar (escribe en un archivo de texto o binario información del momento en el que se ha cumplido la regla y del paquete que lo ha hecho), alertar (además de añadir una entrada en el registro, realiza una acción para notificar a los administradores de la red) y descartar (además de registrar y alertar, descarta el paquete para evitar que llegue a su destino. Esta acción sólo es posible si snort se encuentra instalado en la máquina enrutadora. Si snort se encuentra en una máquina conectada a la red pero no tiene control sobre los paquetes que circulan por la misma, no será capaz de impedir que el paquete llegue hasta su destino). También es posible, mediante un sistema de módulos, añadir acciones propias y emplearlas en las reglas.

3.1.3.1 Reglas para detectar ataques de inyección SQL

Las reglas para detectar ataques de inyección de SQL se centran en detectar en la solicitud HTTP los caracteres necesarios para poder realizar este ataque. La primera regla se centra en caracteres especiales de SQL y de MySQL, como son el delimitador de cadena de texto «'» y el delimitador de comentarios («--» en SQL, «#» en MySQL). Las dos siguientes se centran en detectar una palabra clave en SQL tras un símbolo «=».

3.1.3.2 Reglas para detectar ataques DoS

Las reglas para detectar ataques de este tipo se basan en la capacidad de snort de realizar una cuenta de los paquetes que cumplen las condiciones de una regla antes de realizar la acción requerida. Así, ambas reglas indican a snort que alerte cuando se reciban más de 50 peticiones en 1 segundo desde el mismo origen. La única diferencia entre las reglas es que la primera de ellas detectaría ataques DoS enfocados a los puertos del servidor web, mientras que la segunda detecta ataques usando el protocolo ICMP (ping). Estas reglas no serían capaces de detectar un ataque DDoS pero son suficientes para el presente caso.

3.2 Atacante

Siguiendo la sugerencia de la tutora, se ha optado por el uso de la distribución kali linux, que se ha instalado usando la configuración por defecto en una nueva máquina virtual de 64 bits (versión 1.1.0a amd64). Esta distribución presenta una gran cantidad de herramientas para la monitorización de redes y sistemas y para extraer información de ellos. Si bien para ello se incluyen multitud de herramientas de alto nivel diseñadas para combinar el uso de diversas aplicaciones de bajo nivel en lo que se podría denominar como un ataque combinado o de amplio espectro, se ha optado, en aras de la sencillez, por emplear directamente las aplicaciones de bajo nivel y observar el éxito en la detección de los ataques iniciados.

4 Fase de Ataque

Se han realizado diversas repeticiones de los ataques que se muestran a continuación. También se ha procedido a usar la herramienta OTRS, combinando los accesos como usuario y como administrador.

4.1 Uso normal

Para simular un uso normal, se ha procedido a crear diversas 'colas', agrupaciones que permiten clasificar tickets (por ejemplo, según el profesional que deba atender la incidencia a continuación), a crear diversos tickets en estas colas y a gestionarlos. Las operaciones realizadas en este apartado no han supuesto la activación de ninguna de las alertas configuradas en snort.

4.2 Ataque vulnerabilidades

Para este ataque se ha optado por sqlmap, una herramienta para realizar ataques de inyección de código SQL que permite automatizar la extracción de información de Bases de Datos. En la ejecución del comando, se ha añadido el parámetro '`delay`' para forzar un tiempo de retraso entre consultas consecutivas y evitar confusiones activando la alerta por posible ataque TCP DoS.

El URL seleccionado para el ataque es aquel que permite a un profesional (interfaz `index.pl`) visualizar (`Action=AgentTicketZoom`) la información almacenada sobre un ticket (`TicketId=1`). Ya se ha visto como los ataques de inyección SQL se deben intentar sobre parámetros que parezcan destinados a realizar una consulta sobre Base de Datos. En este caso, la acción parece más el acceso al módulo que realiza la acción, mientras que el identificador de ticket parece que será el elemento comparador. Al ser una aplicación de código libre cuyo código fuente es facilmente descargable y consultable, es posible realizar una análisis pormenorizado previo al ataque identificando posibles módulos que no hayan sido debidamente protegidos.

En ninguno de los ataques sqlmap ha tenido éxito obteniendo información de la Base de Datos donde se almacena la información de OTRS. Esto es así porque la aplicación está bien protegida contra este tipo de ataques (La principal medida de protección contra ataques por inyección de SQL consisten en sanear los parámetros de entrada para comprobar que son del tipo que se desea y no contienen caracteres extraños que puedan iniciar código malicioso).

A continuación se muestra la información producida tras uno de estos ataques:

```
$> sqlmap --delay=0.03 -a -u http://$VICTIM/otrs/index.pl?Action=AgentTicketZoom;TicketID=1
sqlmap/1.0-dev - automatic SQL injection and database takeover tool
    http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is
illegal. It is the end user's responsibility to obey all applicable local, sd federal laws.
Developers assume no liability and are not responsible for any misuse or damage caused by this
program

[*] starting at 19:52:50

[19:52:51] [INFO] testing connection to the target URL
[19:52:51] [INFO] testing if the target URL is stable. This can take a couple of seconds
[19:52:52] [INFO] target URL is stable
[19:52:52] [INFO] testing if GET parameter 'Action' is dynamic
[19:52:52] [WARNING] GET parameter 'Action' does not appear dynamic
[19:52:53] [WARNING] heuristic (basic) test shows that GET parameter 'Action' might not be
injectable
```

```
[19:52:53] [INFO] testing for SQL injection on GET parameter 'Action'
[19:52:53] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[19:52:53] [WARNING] reflective value(s) found and filtering out
[19:52:58] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'
[19:52:59] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[19:53:00] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause'
[19:53:02] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[19:53:04] [INFO] testing 'MySQL inline queries'
[19:53:04] [INFO] testing 'PostgreSQL inline queries'
[19:53:04] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[19:53:05] [INFO] testing 'Oracle inline queries'
[19:53:05] [INFO] testing 'SQLite inline queries'
[19:53:05] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[19:53:07] [INFO] testing 'PostgreSQL > 8.1 stacked queries'
[19:53:08] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries'
[19:53:10] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'
[19:53:12] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[19:53:13] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind'
[19:53:15] [INFO] testing 'Oracle AND time-based blind'
[19:53:16] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
[19:53:32] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[19:53:32] [WARNING] using unescaped version of the test because of zero knowledge of the back-
end DBMS. You can try to explicitly set it using option '--dbms'
[19:53:47] [WARNING] GET parameter 'Action' is not injectable
[19:53:47] [CRITICAL] all tested parameters appear to be not injectable. Try to increase '--
level'/'--risk' values to perform more tests. Also, you can try to rerun by ng either a valid
value for option '--string' (or '--regexp')

[*] shutting down at 19:53:47
```

4.3 Ataque DoS

Para este ataque se han seleccionado dos herramientas, la ubicua ping, que incluye un modo inundación sólo disponible para los administradores de las máquinas, y la herramienta hping3, una aplicación capaz de enviar paquetes TCP/IP. Ambas herramientas son muy sencillas y consiguieron bloquear la conexión de red de la máquina víctima, de tal manera que hasta que no se detuvieron no se pudo volver a acceder a administrarla.

4.3.1 DoS ICMP

La herramienta ping es bien conocida para cualquier operador de sistemas: permite enviar solicitudes ECHO ICMP para determinar si hay comunicación entre dos máquinas. Una serie de parámetros permiten al usuario administrador emplear esta herramienta para inundar a una máquina remota de solicitudes de este tipo.

A continuación se muestra la información producida tras uno de estos ataques:

```
#> ping $VICTIM -i 0 -f -c 1000
PING 192.168.1.40 (192.168.1.40) 56(84) bytes of data.
..
--- 192.168.1.40 ping statistics ---
50 packets transmitted, 48 received, 4% packet loss, time 618ms
rtt min/avg/max/mdev = 3.939/188.519/423.728/112.517 ms, pipe 32, ipg/ewma 12.623/119.167 ms
```

4.3.2 DoS HTTP

La herramienta hping3 permite enviar paquetes TCP/IP y mostrar las respuestas recibidas de una manera similar a la herramienta ping. Una amplia lista de parámetros permiten adaptar el funcionamiento de la herramienta a multitud de propósitos.

A pesar de usar el parámetro '-c' 100, siempre se ha tenido que interrumpir la ejecución de la aplicación. El resto de parámetros se ha seleccionado siguiendo diversas recomendaciones de uso de la herramienta[2].

A continuación se muestra la información producida tras uno de estos ataques:

```
$> hping3 -c 100 -d 120 -S -p 80 --flood $VICTIM

hping3 -c 100 -d 120 -S -p 80 --flood $VICTIM
HPING 192.168.1.40 (eth0 192.168.1.40): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.40 hping statistic ---
5984 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

[2] http://www.blackmoreops.com/2015/04/21/denial-of-service-attack-dos-using-hping3-with-spoofed-ip-in-kali-linux/

5 Fase de Análisis

Snort produce registros en modo texto y en modo binario, guardando los paquetes que generaron las alertas. El modo texto se puede abrir con cualquier editor de texto, mientras que el binario requiere un visor de paquetes (el más popular, con muchas funcionalidades y gratuito, es Wireshark). A continuación se muestra una alerta en ambos formatos[3] para cada uno de los ataques realizados.

```
[**] [1:1000001:1] SQL Injection - Comments and text
delimiter [**]
[Priority: 0]
05/26-19:47:59.231924 192.168.1.136:36188 -> 192.168.1.40:80
TCP TTL:64 TOS:0x0 ID:18229 IpLen:20 DgmLen:463 DF
***AP*** Seq: 0x671A65BA  Ack: 0xCFC6CE68  Win: 0x1C9
TcpLen: 32
TCP Options (3) => NOP NOP TS: 5120653 1289379
```

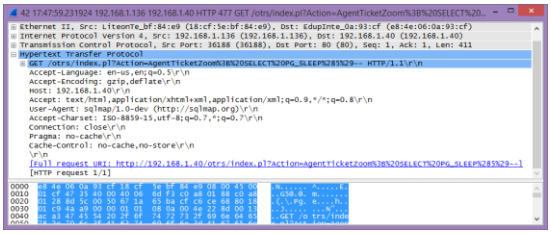


Ilustración 7. Muestra de alerta por ataque de inyección SQL

```
[**] [1:1000003:1] Possible TCP DoS [**]
[Priority: 0]
05/26-19:24:12.923065 192.168.1.136:4267 -> 192.168.1.40:80
TCP TTL:64 TOS:0x0 ID:6292 IpLen:20 DgmLen:160
******S* Seq: 0x2BB7117F  Ack: 0x1BDAAF9  Win: 0x200
TcpLen: 20
```

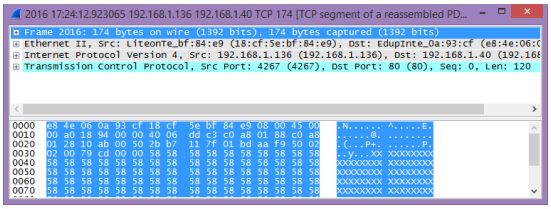


Ilustración 8. Muestra de alerta por ataque TCP DoS

```
 [**] [1:1000004:1] Possible ICMP DoS [**]
[Priority: 0]
05/26-19:18:56.289594 192.168.1.136 -> 192.168.1.40
ICMP TTL:64 TOS:0x0 ID:15607 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:14264  Seq:82  ECHO
```

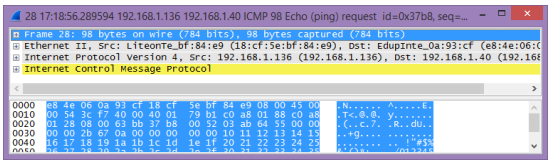


Ilustración 9. Muestra de alerta por ataque ICMP DoS

No se han observado activaciones de las reglas de snort tras uso normal del aplicativo instalado. Teniendo en cuenta las reglas definidas, esto podría haber pasado en alguno de los siguientes casos:

1. Puesto que los navegadores modernos intentan realizar la carga de contenido estático (imágenes, estilos, código ejecutable) en paralelo, la presencia de alguna página con abundante contenido de este tipo podría suponer que se alcanzara el umbral de peticiones HTTP por segundo desde un mismo origen definidas en la regla contra el DoS HTTP;
2. Si algún elemento importante de la estructura de información de OTRS (una cola, por ejemplo) contuviera alguna palabra clave SQL en parte de su nombre (ANDalucía, por ejemplo) y este elemento pudiera referirse por su nombre en el URL (Action=AdminQueue&QueueName=andalucía), se produciría un falso positivo de la regla de detección de inyección sql, si bien en el transcurso de las pruebas no se ha encontrado ningún elemento referenciable por el nombre.

[3] Las horas mostradas en las capturas de Wireshark son en UTC, mientras que las de los registros de texto son en hora local (UTC+2)

6 Conclusión

El presente proyecto esquematiza el uso de snort para detectar tráfico no deseado en un entorno de red simplificado: muestra la instalación del aplicativo principal y de snort como herramienta de detección de intrusos; muestra la configuración necesaria de snort para detectar una serie de ataques previamente definidos; muestra cómo realizar estos ataques desde una máquina exterior y, finalmente, muestra la información producida por snort sobre los ataques realizados.

Es necesario hacer hincapié en que snort únicamente detecta patrones: no es capaz de detectar ataques para los que no tenga una regla de detección configurada. Un interesante ejercicio sería repetir el proyecto con un IDS de detección de anomalías, capaz de detectar intrusos para los que no tiene un patrón preestablecido, o con un IPS, capaz de reaccionar ante los intrusos detectados.

7 Tabla de Ilustraciones

8 Bibliografía

[1] «Official Kali Linux Documentation,» [En línea]. Available: https://www.kali.org/kali-linux-documentation/. [Último acceso: 01 04 2015].

[2] «hping3(8) - Linux man page,» [En línea]. Available: http://linux.die.net/man/8/hping3. [Último acceso: 05 04 2015].

[3] «ping(8) - Linux man page,» [En línea]. Available: http://linux.die.net/man/8/ping. [Último acceso: 05 04 2015].

[4] «Snort Documentation,» [En línea]. Available: https://www.snort.org/#documents. [Último acceso: 22 03 2015].

[5] SANS Institute, «Choosing an Intrusion Detection System that Best Suits your Organization,» 2002.

[6] P. N. Raju, «State-of-the-art Intrusion Detection: Technologies, Challenges, and Evaluation,» 2005.

9 Anexo

9.1 Reglas de snort

9.1.1 web-attacks.rules

```
alert tcp any any -> $LOCAL $HTTP_PORTS (msg:"SQL Injection - Comments and text delimiter";
flow:to_server;pcre:"/(\%27)|(\')|(\-\-)|(%23)|(#)/i"; sid:1000001; rev:1;)

alert tcp any any -> $LOCAL $HTTP_PORTS (msg:"SQL Injection - SQL key word";
flow:to_server;pcre:"/((\%3D)|(=)).*((\%6F)|o|(\%4F))((\%72)|r|(\%52))/i"; sid:1000002; rev:1;)

alert tcp any any -> $LOCAL $HTTP_PORTS (msg:"SQL Injection - SQL key word";
flow:to_server;pcre:"/((\%3D)|(=)).*((\%61)|a|(\%41))((\%6E)|n|(\%4E))((\%64)|d|(\%44))/i";
sid:1000005; rev:1;)
```

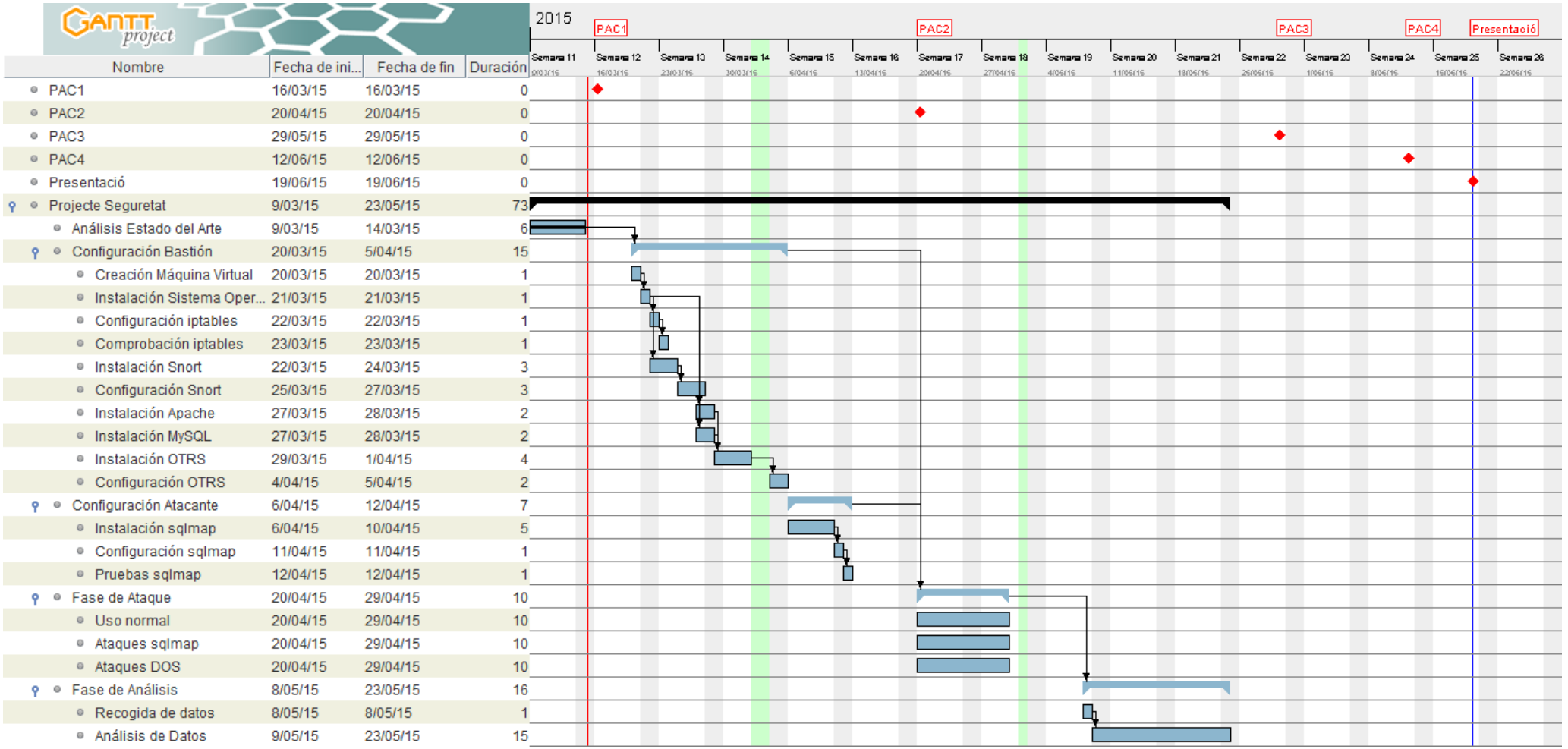
9.1.2 ddos.rules

```
alert tcp any any -> $LOCAL $HTTP_PORTS (msg:"Possible TCP DoS"; flow: to_server; flags: S;
detection_filter: track by_src, count 50, seconds 1; sid:1000003;rev:1;)

alert ICMP any any -> $LOCAL any (msg:"Possible ICMP DoS"; detection_filter: track by_src, count
50, seconds 1; sid:1000004; rev:1; )
```

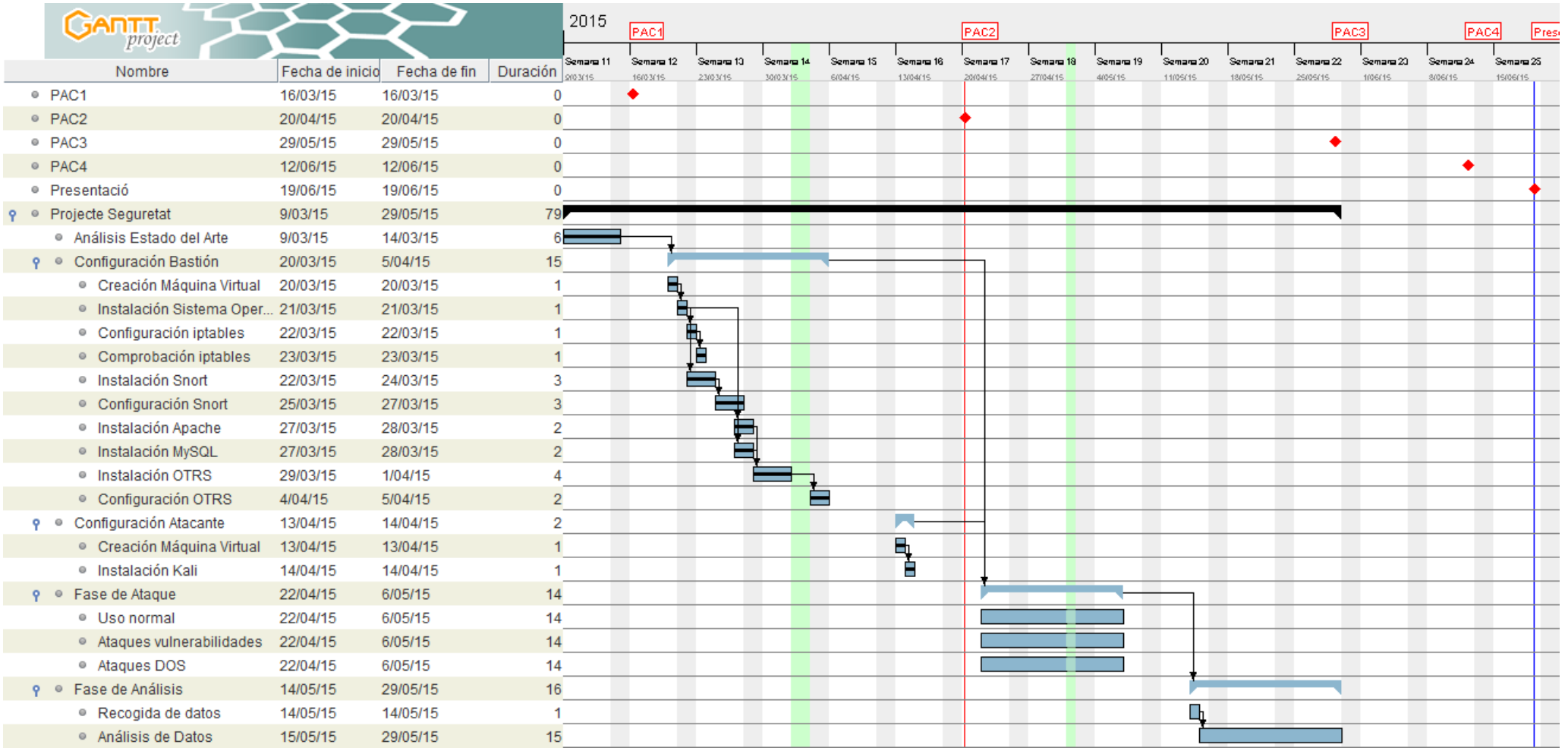
9.2 Diagrama de Gantt

9.2.1 PAC1



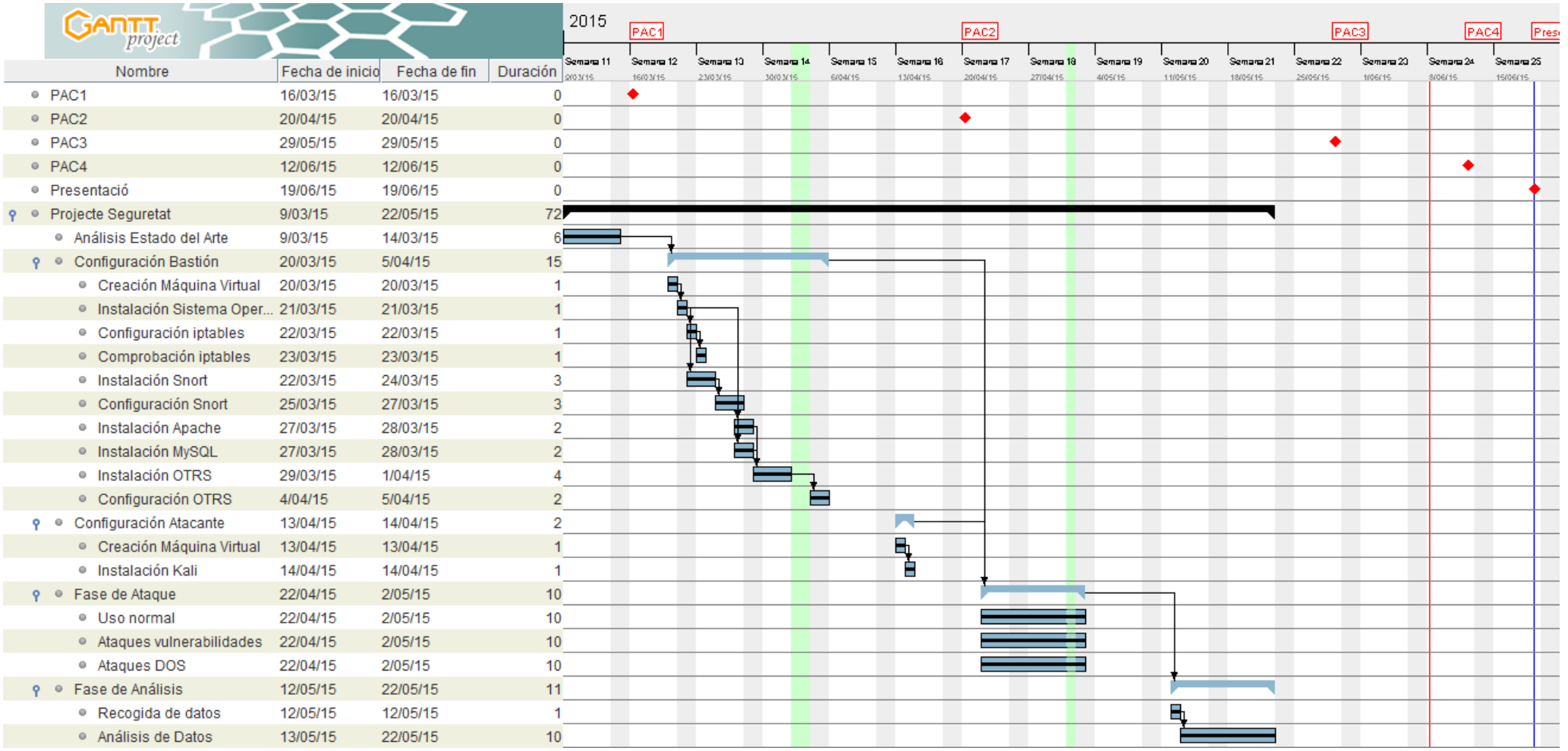
Nombre	Fecha de ini...	Fecha de fin	Duración
PAC1	16/03/15	16/03/15	0
PAC2	20/04/15	20/04/15	0
PAC3	29/05/15	29/05/15	0
PAC4	12/06/15	12/06/15	0
Presentació	19/06/15	19/06/15	0
Projecte Seguretat	9/03/15	23/05/15	73
Análisis Estado del Arte	9/03/15	14/03/15	6
Configuración Bastión	20/03/15	5/04/15	15
Creación Máquina Virtual	20/03/15	20/03/15	1
Instalación Sistema Oper...	21/03/15	21/03/15	1
Configuración iptables	22/03/15	22/03/15	1
Comprobación iptables	23/03/15	23/03/15	1
Instalación Snort	22/03/15	24/03/15	3
Configuración Snort	25/03/15	27/03/15	3
Instalación Apache	27/03/15	28/03/15	2
Instalación MySQL	27/03/15	28/03/15	2
Instalación OTRS	29/03/15	1/04/15	4
Configuración OTRS	4/04/15	5/04/15	2
Configuración Atacante	6/04/15	12/04/15	7
Instalación sqlmap	6/04/15	10/04/15	5
Configuración sqlmap	11/04/15	11/04/15	1
Pruebas sqlmap	12/04/15	12/04/15	1
Fase de Ataque	20/04/15	29/04/15	10
Uso normal	20/04/15	29/04/15	10
Ataques sqlmap	20/04/15	29/04/15	10
Ataques DOS	20/04/15	29/04/15	10
Fase de Análisis	8/05/15	23/05/15	16
Recogida de datos	8/05/15	8/05/15	1
Análisis de Datos	9/05/15	23/05/15	15

9.2.2 PAC2



Nombre	Fecha de inicio	Fecha de fin	Duración
PAC1	16/03/15	16/03/15	0
PAC2	20/04/15	20/04/15	0
PAC3	29/05/15	29/05/15	0
PAC4	12/06/15	12/06/15	0
Presentació	19/06/15	19/06/15	0
Projecte Seguretat	9/03/15	29/05/15	79
Análisis Estado del Arte	9/03/15	14/03/15	6
Configuración Bastión	20/03/15	5/04/15	15
Creación Máquina Virtual	20/03/15	20/03/15	1
Instalación Sistema Oper...	21/03/15	21/03/15	1
Configuración iptables	22/03/15	22/03/15	1
Comprobación iptables	23/03/15	23/03/15	1
Instalación Snort	22/03/15	24/03/15	3
Configuración Snort	25/03/15	27/03/15	3
Instalación Apache	27/03/15	28/03/15	2
Instalación MySQL	27/03/15	28/03/15	2
Instalación OTRS	29/03/15	1/04/15	4
Configuración OTRS	4/04/15	5/04/15	2
Configuración Atacante	13/04/15	14/04/15	2
Creación Máquina Virtual	13/04/15	13/04/15	1
Instalación Kali	14/04/15	14/04/15	1
Fase de Ataque	22/04/15	6/05/15	14
Uso normal	22/04/15	6/05/15	14
Ataques vulnerabilidades	22/04/15	6/05/15	14
Ataques DOS	22/04/15	6/05/15	14
Fase de Análisis	14/05/15	29/05/15	16
Recogida de datos	14/05/15	14/05/15	1
Análisis de Datos	15/05/15	29/05/15	15

9.2.3 PAC3


GANTT project
2015
PAC1
PAC2
PAC3
PAC4
Nombre
Fecha de inicio
Fecha de fin
Duración
PAC1 16/03/15 16/03/15 0
PAC2 20/04/15 20/04/15 0
PAC3 29/05/15 29/05/15 0
PAC4 12/06/15 12/06/15 0
Presentació 19/06/15 19/06/15 0
Projecte Seguretat 9/03/15 22/05/15 72
Análisis Estado del Arte 9/03/15 14/03/15 6
Configuración Bastión 20/03/15 5/04/15 15
Creación Máquina Virtual 20/03/15 20/03/15 1
Instalación Sistema Oper... 21/03/15 21/03/15 1
Configuración iptables 22/03/15 22/03/15 1
Comprobación iptables 23/03/15 23/03/15 1
Instalación Snort 22/03/15 24/03/15 3
Configuración Snort 25/03/15 27/03/15 3
Instalación Apache 27/03/15 28/03/15 2
Instalación MySQL 27/03/15 28/03/15 2
Instalación OTRS 29/03/15 1/04/15 4
Configuración OTRS 4/04/15 5/04/15 2
Configuración Atacante 13/04/15 14/04/15 2
Creación Máquina Virtual 13/04/15 13/04/15 1
Instalación Kali 14/04/15 14/04/15 1
Fase de Ataque 22/04/15 2/05/15 10
Uso normal 22/04/15 2/05/15 10
Ataques vulnerabilidades 22/04/15 2/05/15 10
Ataques DOS 22/04/15 2/05/15 10
Fase de Análisis 12/05/15 22/05/15 11
Recogida de datos 12/05/15 12/05/15 1
Análisis de Datos 13/05/15 22/05/15 10