# ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL


**Facultad de Ingeniería en Electricidad y Computación**

"ANÁLISIS DE VULNERABILIDADES, PENTESTING Y ACCIONES CORRECTIVAS SOBRE EL SISTEMA WEB DE COMISIONES UTILIZADO POR INSTITUCIONES FINANCIERAS"

**EXAMEN DE GRADO (COMPLEXIVO)**

Previo a la obtención del grado de:

**MAGÍSTER EN SEGURIDAD INFORMÁTICA APLICADA**

ALEXANDER XAVIER TOALA PAZ

GUAYAQUIL – ECUADOR
AÑO 2017

# AGRADECIMIENTO

Gracias a Dios y a mis padres, por el apoyo incondicional y su constante esfuerzo en brindarme una excelente educación. A mi bella esposa Andrea Becerra, por alentarme en todo momento y su infinito amor. Y a cada uno de los docentes que impartieron sus conocimientos.

# DEDICATORIA

El siguiente trabajo lo dedico a Dios, a mis padres, a mi abuelita, a mi esposa y preciosa hija. Por ellos soy quien soy.

# TRIBUNAL DE SUSTENTACIÓN

| | |
|---|---|
| **MSIG Lenin Freire** | **MSIG Juan Carlos García** |
| PROFESOR DELEGADO | PROFESOR DELEGADO |
| POR LA SUBDECANA DE LA FIEC | POR LA SUBDECANA DE LA FIEC |

# RESUMEN

El Sistema Web o Motor de Comisiones se encarga de administrar de manera automatizada los procesos involucrados en la generación de las comisiones para varios productos transaccionales correspondientes a entidades financieras. Existen varios módulos que constituyen el sistema, tales como la aplicación Web, fuentes de datos, Motor de Reglas, Motor de Cálculo, directorio de seguridad, entre otros.

Se analizará la seguridad de todos los hosts y componentes que conforman la aplicación Web, Motor de Reglas y directorio de seguridad. Para lograr esto, se realizará un hacking ético interno en la modalidad White-box-hacking con las fases de escaneo, análisis de vulnerabilidades y pentesting. También se gestionará la calidad del código fuente con la herramienta OWASP SonarQube [14].

Como parte final, se incluirán resultados del pentesting y acciones preventivas / correctivas.

# ÍNDICE GENERAL

# ABREVIATURAS Y SIMBOLOGÍA

| | |
|---|---|
| BRMS | Business Rules Management System. |
| OWASP | The Open Web Application Security Project. |
| JDK | Java Development Kit. |
| LDAP | Lightweight Directory Access Protocol. |
| LAN | Local Area Network. |
| OVB | Oracle Virtual Box. |
| SSH | Secure Shell. |
| SSL | Secure Sockets Layer. |
| HTTP | Hypertext Transfer Protocol. |
| DCE | Distributed Computing Environment. |
| MSRPC | Microsoft Remote Procedure Call. |
| TCP | Transmission Control Protocol. |
| OSVDB | Open Source Vulnerability Database. |
| TNS | Transparent Network Substrate. |
| SMB | Server Message Block. |
| DoS | Denial of Service. |
| UFW | Uncomplicated Firewall. |
| WebDAV | Web Distributed Authoring and Versioning. |
| IPS / IDS | Intrusion Prevention System / Intrusion Detection System. |
| WAF | Web Application Firewall. |

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# INTRODUCCIÓN

El presente trabajo describe las fases de un hacking ético, iniciando con el escaneo, análisis de vulnerabilidades y pentesting a hosts y componentes que conforman la aplicación Web, Motor de Reglas y directorio de seguridad del sistema Web o Motor de Comisiones utilizado por instituciones financieras.

El capítulo 1 detalla la descripción del problema y la solución propuesta.

El capítulo 2 profundiza en el modelo del negocio, arquitectura del software, plataforma, el hacking de caja blanca y una revisión de calidad del código fuente de la aplicación Web.

El capítulo 3 incluye los resultados de exploits realizados y acciones de prevención / corrección al sistema de Comisiones.

# CAPÍTULO 1

# GENERALIDADES

## 1.1 DESCRIPCIÓN DEL PROBLEMA

El Sistema Web o Motor de Comisiones se encarga de administrar de manera automatizada los procesos involucrados en la generación de las comisiones para varios productos tales como Cuenta Corriente, Cuenta de Ahorro, Cuenta Vista, Avance de Sueldo, etc., correspondientes a entidades financieras.

Se requiere analizar la seguridad de todos los hosts y componentes que conforman la aplicación Web, Motor de Reglas y directorio de seguridad. Todo esto en ambiente de desarrollo.

## 1.2 SOLUCIÓN PROPUESTA

La presente propuesta comprende el alcance de los siguientes puntos:

- Realizar escaneo, análisis de vulnerabilidades y exploits en hosts y componentes de la aplicación Web, motor de Reglas y directorio de seguridad.
- Implementación de Owasp SonarQube aplicado al código fuente de la aplicación Web.
- Toma de acciones preventivas / correctivas de los componentes del sistema.

Esto permitirá entregar los niveles de seguridad y escalabilidad que se necesitan para soportar los requerimientos del negocio.

# CAPÍTULO 2

# METODOLOGÍA Y DESARROLLO DE LA SOLUCIÓN

## 2.1 MODELO DEL NEGOCIO

### Administración de Comisiones

Comisión es la cantidad que se cobra por realizar transacciones comerciales que corresponden a un porcentaje sobre el importe de operación.

Como parte del negocio de entidades financieras, existen reglas asociadas para establecer el valor de las comisiones que se generan a partir de la utilización de productos tales como Cuenta Corriente, Cuenta de Ahorro, Cuenta Vista, Avance de Sueldo, etc.

En las figuras 2.1 y 2.2 observamos ejemplos de reglas asociadas a productos y también como una regla específica puede cambiar en un determinado tiempo.

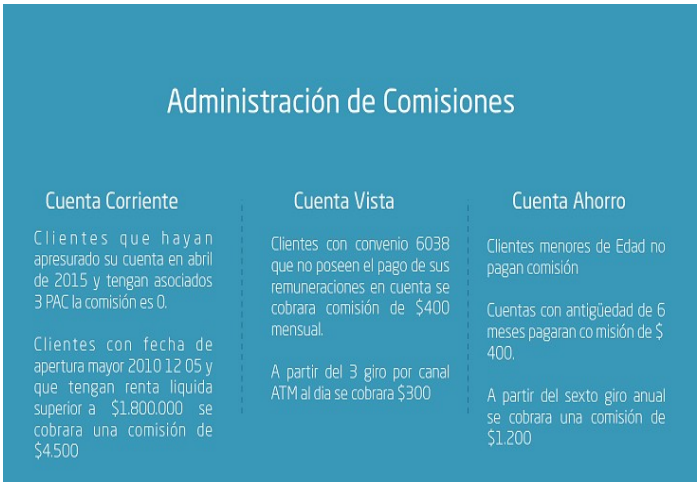



Figura 2.1 Reglas asociadas a productos

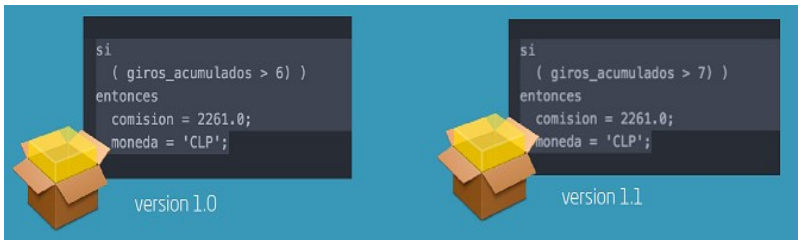



Figura 2.2 Regla específica en dos versiones

El ciclo de negocio del sistema de Comisiones está compuesto por el área comercial quien define las reglas, y el área operacional quien implementa los diversos módulos para gestionar las reglas y cobro de comisiones.


Figura 2.3 Ciclo del negocio

## 2.2 ARQUITECTURA DEL SOFTWARE

La arquitectura del proyecto Sistema de Comisiones está planteada utilizando una arquitectura en capas, sostenidas sobre varias tecnologías, las que, a su vez, exponen sus funcionalidades a través de entornos web. Existen varios módulos que constituyen el sistema. A continuación un diagrama general.

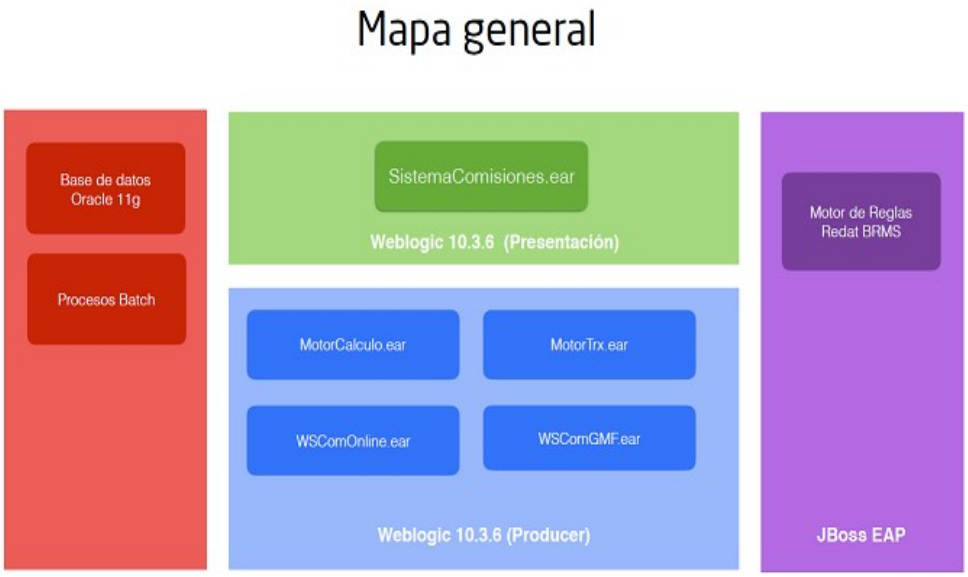



Figura 2.4 Mapa general del Sistema de Comisiones

A continuación, se muestran los componentes de la aplicación Web (Capa de Presentación).

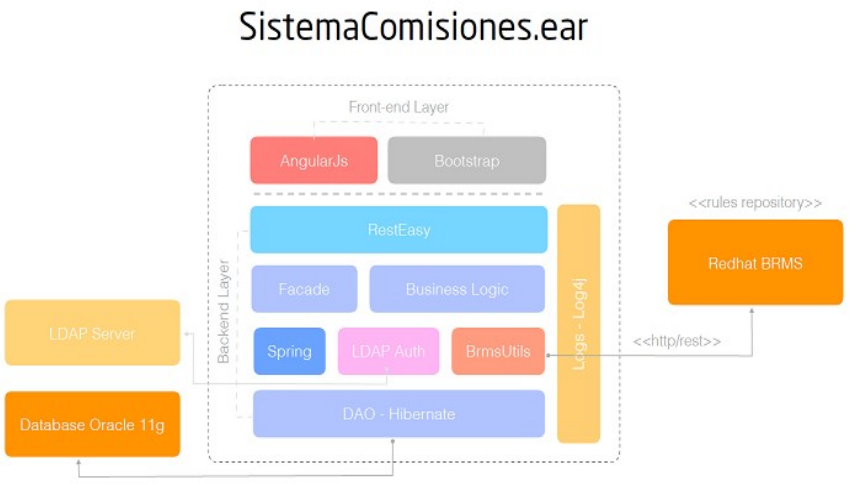



Figura 2.5 Componentes de la aplicación Web

Como principales componentes de la aplicación Web tenemos la base de datos Oracle, servidor WebLogic, servidor Web JBoss, directorio de seguridad LDAP, RedHat BRMS, AngularJS, Spring Framework, Hibernate y Maven, el cual es un software para la gestión y construcción de proyectos Java.

La aplicación Web permite explotar las diferentes funcionalidades del Sistema de Comisiones. El usuario, según sea su perfil, podrá administrar los procesos involucrados en la generación de las comisiones para los clientes de la institución financiera.

Además, el usuario podrá realizar el mantenimiento de los módulos de la aplicación, entre los que se destacan: Planes, Reglas, Productos, Variables, Parámetros, Acceso al Sistema, etc.

Se indica la URL para acceder a la aplicación Web:

**http://{Host}:{Port}/SistemaComisiones**

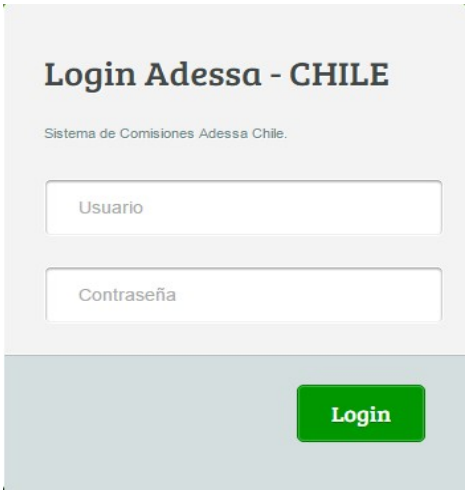



Figura 2.6 Login

En esta pantalla se solicitará las credenciales de ingreso al Sistema; es decir usuario y contraseña (Ver Figura 2.6). La función de esta pantalla es básicamente la de garantizar que todos los usuarios que accedan al sistema estén creados en el mismo. Por defecto el sistema cuenta con un usuario Administrador, el mismo que podrá crear todos los tipos de usuarios requeridos por el cliente. En cuanto a la gestión de usuarios, esta pantalla distinguirá cuatro niveles o perfiles de acceso que son los siguientes: Administrador, Comercial, Autorizador y Operador. Cada nivel de acceso tendrá asignadas funcionalidades distintas. La autenticación es implementada con el directorio de seguridad Open LDAP [3] en ambiente de desarrollo.

El perfil Operador tiene como principal funcionalidad la creación de Planes, quienes contienen uno o más conceptos, y un concepto contiene las reglas del negocio.

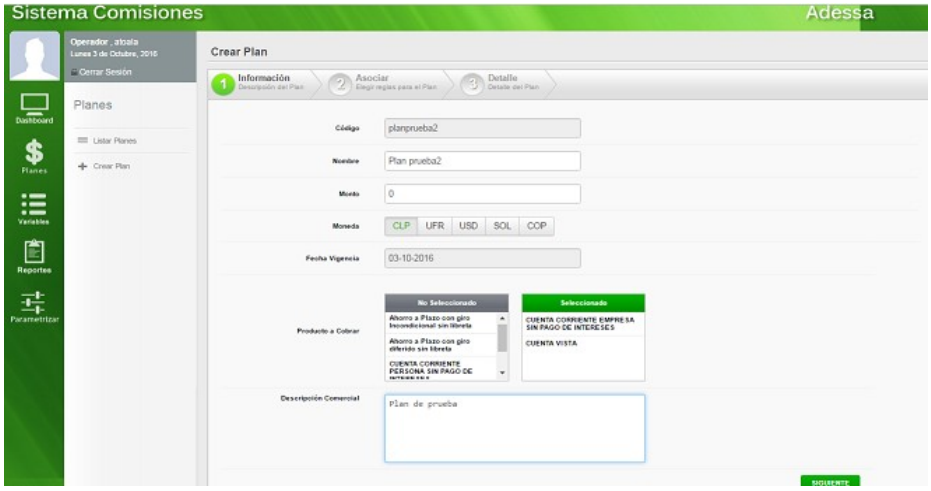



Figura 2.7 Perfil Operador – Creación de Plan

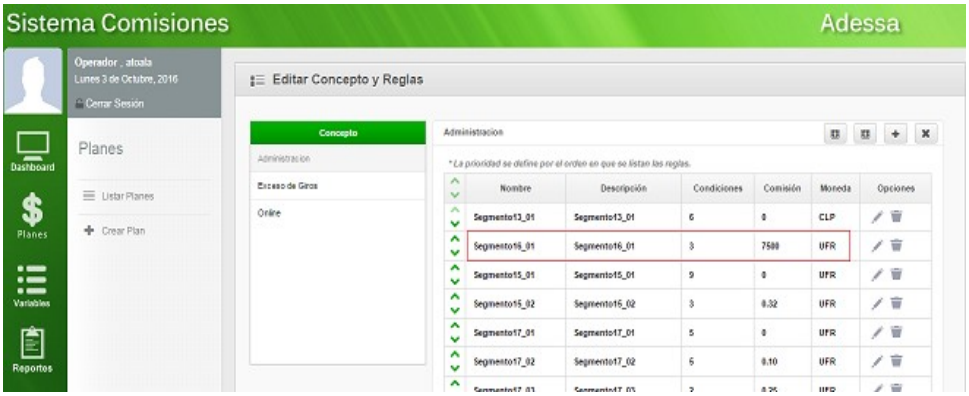



Figura 2.8 Perfil Operador – Listado de Reglas de Negocio

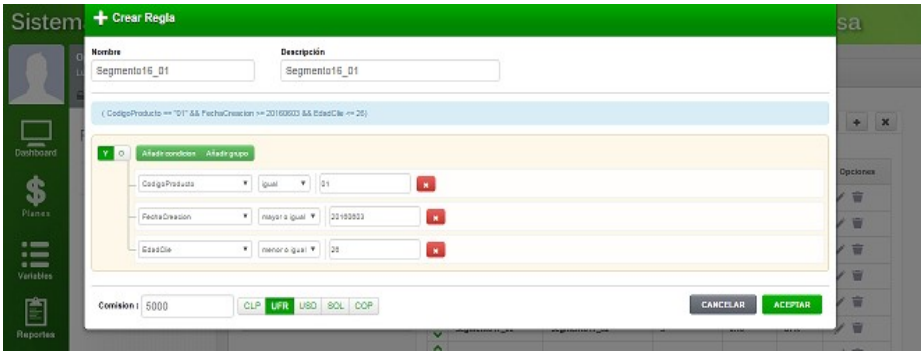



Figura 2.9 Perfil Operador – Creación de Regla

El perfil Autorizador es el encargado de aprobar o no la habilitación/deshabilitación de un Plan creado por el Operador. En la figura 2.10 observamos una solicitud pendiente indicando habilitar el plan Corriente, si el Autorizador lo aprueba, se genera una conexión con el motor de Reglas BRMS (Figura 2.11) donde se alojará el plan con sus respectivas reglas. Dichas reglas luego serán consumidas por el Motor de Cálculo como parte del ciclo del Sistema de Comisiones.

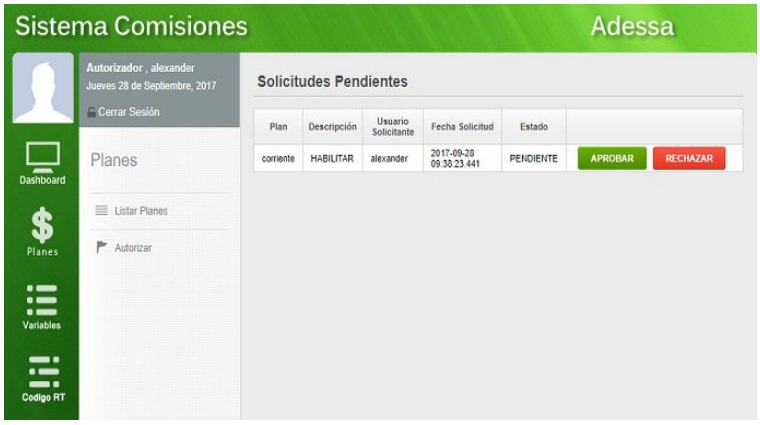



Figura 2.10 Perfil Autorizador – Solicitud para habilitar un Plan

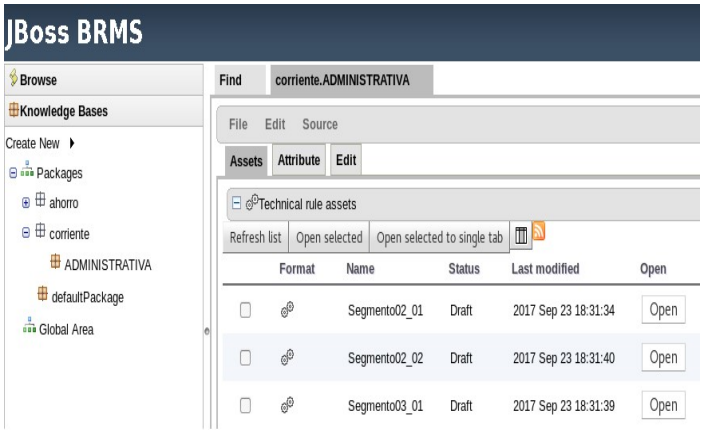



Figura 2.11 Plan Corriente en BRMS

**Herramientas de Desarrollo**

A continuación, se enumeran las herramientas utilizadas para el desarrollo y ejecución de la aplicación Web del Sistema de Comisiones.

**Servidores de Aplicaciones**

Oracle WebLogic Server 10.3.6 (JDK 1.6.0)

JBoss Web Server (JDK 1.7.0)

**Cliente Base de Datos**

Oracle XE Client - 32 bits

**IDE de Desarrollo**

Eclipse Kepler

**Motor de Reglas de Negocio**

BRMS 5.3.1

**Directorio de Seguridad** Open LDAP

## 2.3 DESCRIPCIÓN DE ESCENARIO Y PLATAFORMA

Se realizará un hacking ético interno en la modalidad White-box-hacking [1]; contamos con 2 máquinas físicas y máquinas virtuales (establecidas en una LAN en mi hogar) con todos los componentes instalados que se necesitan para correr la aplicación Web. A continuación, se enumeran las características de las máquinas:

Tabla 1 Características máquina física 1

| | |
|---|---|
| **HOST NAME** | NEO69 |
| **S.O.** | Windows 10 Pro – 64 bits |
| **PROCESADOR - RAM - CPU** | Intel Core i7-7700HQ 7th Gen 16GB Ram - 3.20Ghz |
| **SOFTWARE EN WIN 10** | Oracle WebLogic Server 10.3.6<br>JDK 1.6<br>Eclipse Kepler<br>Oracle DB 11g Release 2<br>SQL Developer |
| **IP** | 192.168.0.17 |
| **MÁQUINA VIRTUAL - IP** | Kali Linux 4.6.0 (OVB)<br>192.168.0.19 |

En el host NEO69 con Windows 10 se encuentra corriendo el servidor WebLogic con la aplicación Web del Sistema de Comisiones (Figura 2.12). Y utilizaremos la máquina virtual con Kali Linux (.19) para realizar el hacking ético a la aplicación y hosts.

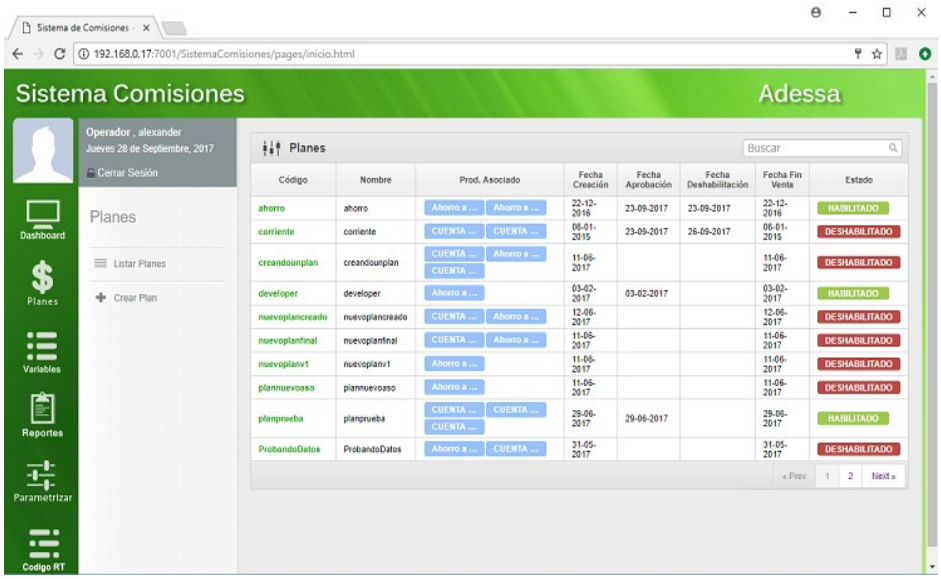



Figura 2.12 Aplicación Web

Tabla 2 Características máquina física 2

| | |
|---|---|
| **HOST NAME** | ALEX |
| **S.O.** | Windows 10 Pro – 64 bits |
| **PROCESADOR - RAM - CPU** | Intel Core i5-4210U 4th Gen - 8GB Ram - 2.20Ghz |
| **MÁQUINA VIRTUAL - IP** | Kali Linux 4.6.0 (OVB) |

| | |
|---|---|
| | 192.168.0.22 |
| **SOFTWARE EN KALI** | BRMS 5.3.1<br>OpenLDAP 2.3 |

En el host ALEX corre una máquina virtual en Kali donde están instalados 2 componentes necesarios para el correcto funcionamiento de la aplicación Web: BRMS y OpenLDAP.

## 2.4 ESCANEO

Utilizamos la herramienta Zenmap para identificar y comprobar los puertos abiertos, servicios levantados y sistema operativo en el host NEO69 y host ALEX (Kali) con el siguiente comando de modo agresivo:

```
nmap -T4 -A -v 192.168.0.17 192.168.0.22
```

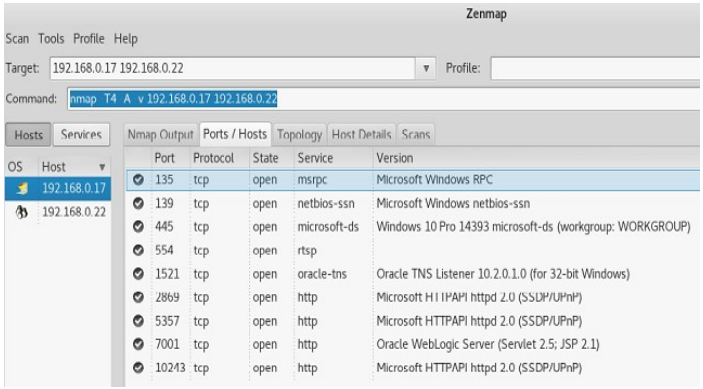



Figura 2.13 Puertos abiertos y servicios en host NEO69

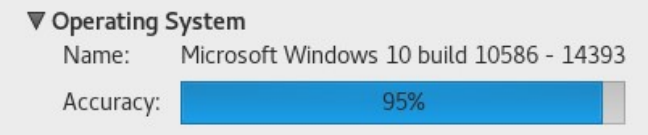



Figura 2.14 Detección S.O. en host NEO69

En la figura 2.13 observamos que el servidor WebLogic está corriendo en el puerto 7001, el cual aloja la aplicación Web de Comisiones.

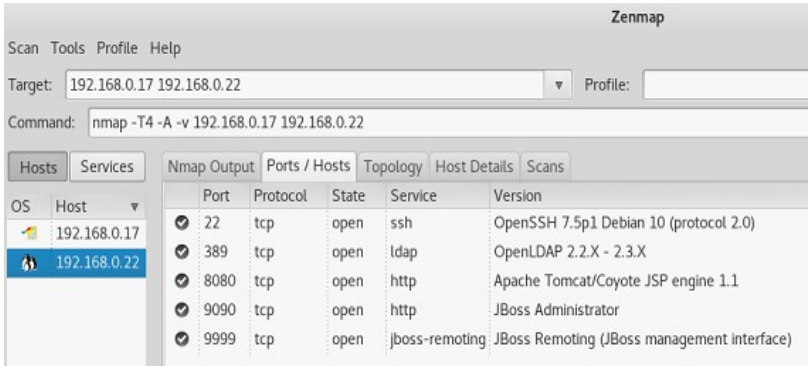



Figura 2.15 Puertos abiertos y servicios en host ALEX (Kali)

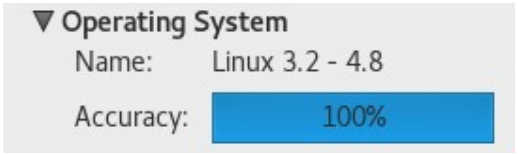



Figura 2.16 Detección Sistema Operativo en host ALEX (Kali)

Y en la figura 2.15 observamos que el servidor Web Apache Tomcat/Coyote está corriendo en el puerto 8080, quien aloja la consola Web del BRMS. También en el puerto 22 el servicio OpenSSH [6].

## 2.5 ANÁLISIS DE VULNERABILIDADES

### 2.5.1 Análisis de Vulnerabilidades en Hosts con OpenVas y Nessus

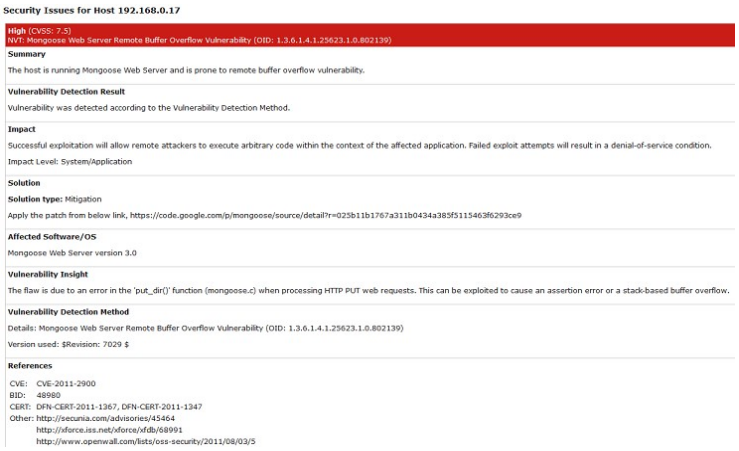
- **OpenVas**

  Utilizamos la herramienta OpenVas [4] para efectuar el análisis en el host NEO69 y host ALEX (Kali). Se hizo empleo del Scan: Full and very deep ultimate.

  **Host NEO69 192.168.0.17**

  En el host se encuentra instalado el servidor WebLogic y la base de datos Oracle 11g.

  Se encontró la siguiente vulnerabilidad con Riesgo Alto:



**Security Issues for Host 192.168.0.17**

**High** (CVSS: 7.5)
NVT: Mongoose Web Server Remote Buffer Overflow Vulnerability (OID: 1.3.6.1.4.1.25623.1.0.802139)

**Summary**

The host is running Mongoose Web Server and is prone to remote buffer overflow vulnerability.

**Vulnerability Detection Result**

Vulnerability was detected according to the Vulnerability Detection Method.

**Impact**

Successful exploitation will allow remote attackers to execute arbitrary code within the context of the affected application. Failed exploit attempts will result in a denial-of-service condition.

Impact Level: System/Application

**Solution**

**Solution type:** Mitigation

Apply the patch from below link, https://code.google.com/p/mongoose/source/detail?r=025b11b1767a311b0434a385f5115463f6293ce9

**Affected Software/OS**

Mongoose Web Server version 3.0

**Vulnerability Insight**

The flaw is due to an error in the 'put_dir()' function (mongoose.c) when processing HTTP PUT web requests. This can be exploited to cause an assertion error or a stack-based buffer overflow.

**Vulnerability Detection Method**

Details: Mongoose Web Server Remote Buffer Overflow Vulnerability (OID: 1.3.6.1.4.1.25623.1.0.802139)

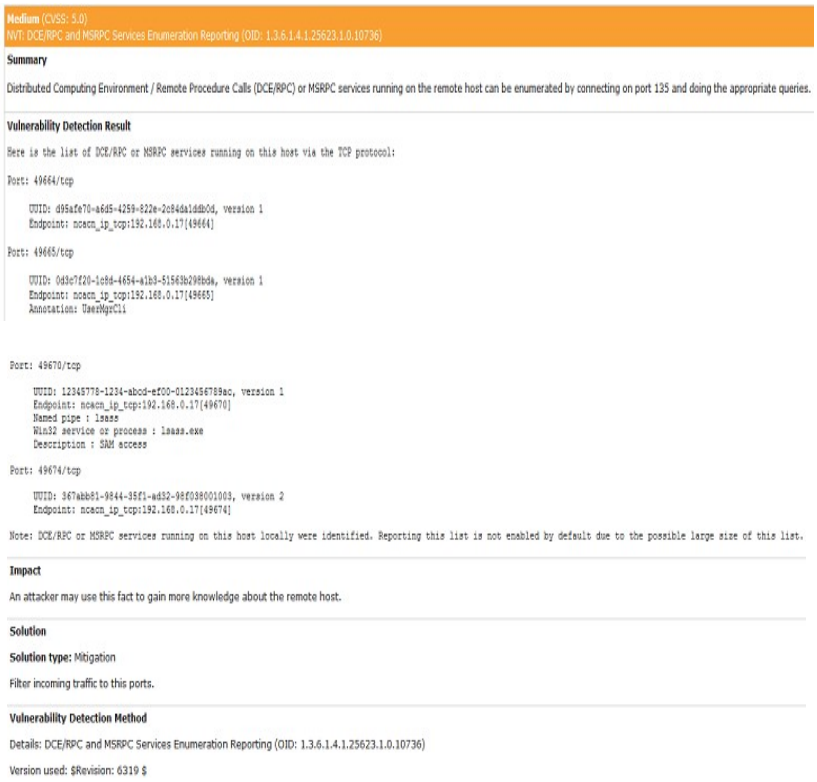
Version used: $Revision: 7029 $

**References**

CVE: CVE-2011-2900
BID: 48980
CERT: DFN-CERT-2011-1367, DFN-CERT-2011-1347
Other: http://secunia.com/advisories/45464
http://xforce.iss.net/xforce/xfdb/68991
http://www.openwall.com/lists/oss-security/2011/08/03/5

Figura 2.17 Vulnerabilidad con Riesgo Alto en host NEO69

Y la siguiente vulnerabilidad con Riesgo Medio:



**Medium** (CVSS: 5.0)
NVT: DCE/RPC and MSRPC Services Enumeration Reporting (OID: 1.3.6.1.4.1.25623.1.0.10736)

**Summary**

Distributed Computing Environment / Remote Procedure Calls (DCE/RPC) or MSRPC services running on the remote host can be enumerated by connecting on port 135 and doing the appropriate queries.

**Vulnerability Detection Result**

Here is the list of DCE/RPC or MSRPC services running on this host via the TCP protocol:

Port: 49664/tcp

UUID: d95afe70-a6d5-4259-822e-2c84da1ddb0d, version 1
Endpoint: ncacn_ip_tcp:192.168.0.17[49664]

Port: 49665/tcp

UUID: 0d3c7f20-1c8d-4654-a1b3-51563b298bda, version 1
Endpoint: ncacn_ip_tcp:192.168.0.17[49665]
Annotation: UserMgrCli

Port: 49670/tcp

UUID: 12345778-1234-abcd-ef00-0123456789ac, version 1
Endpoint: ncacn_ip_tcp:192.168.0.17[49670]
Named pipe : lsass
Win32 service or process : lsass.exe
Description : SAM access

Port: 49674/tcp

UUID: 367abb81-9844-35f1-ad32-98f038001003, version 2
Endpoint: ncacn_ip_tcp:192.168.0.17[49674]

Note: DCE/RPC or MSRPC services running on this host locally were identified. Reporting this list is not enabled by default due to the possible large size of this list.

**Impact**

An attacker may use this fact to gain more knowledge about the remote host.

**Solution**

**Solution type:** Mitigation

Filter incoming traffic to this ports.

**Vulnerability Detection Method**

Details: DCE/RPC and MSRPC Services Enumeration Reporting (OID: 1.3.6.1.4.1.25623.1.0.10736)

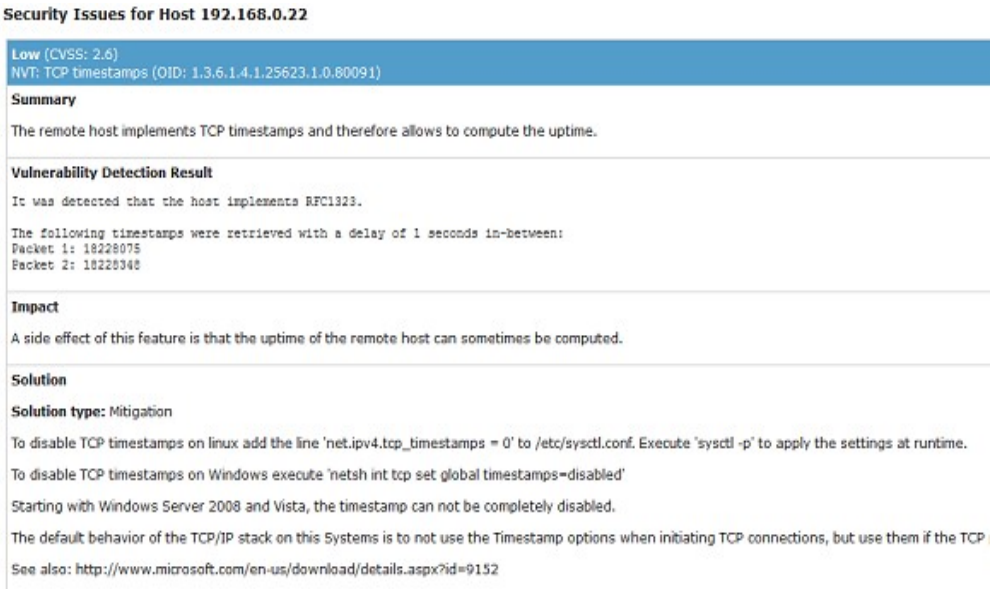
Version used: $Revision: 6319 $

Figura 2.18 Vulnerabilidad con Riesgo Medio en host NEO69

**Host ALEX (Kali) 192.168.0.22**

En este host se encuentra instalado el BRMS y OpenLDAP.

Se encontró 1 vulnerabilidad de Riesgo Bajo:



**Security Issues for Host 192.168.0.22**

**Low** (CVSS: 2.6)
NVT: TCP timestamps (OID: 1.3.6.1.4.1.25623.1.0.80091)

**Summary**

The remote host implements TCP timestamps and therefore allows to compute the uptime.

**Vulnerability Detection Result**

It was detected that the host implements RFC1323.

The following timestamps were retrieved with a delay of 1 seconds in-between:
Packet 1: 18228075
Packet 2: 18228348

**Impact**

A side effect of this feature is that the uptime of the remote host can sometimes be computed.

**Solution**

**Solution type:** Mitigation

To disable TCP timestamps on linux add the line 'net.ipv4.tcp_timestamps = 0' to /etc/sysctl.conf. Execute 'sysctl -p' to apply the settings at runtime.

To disable TCP timestamps on Windows execute 'netsh int tcp set global timestamps=disabled'

Starting with Windows Server 2008 and Vista, the timestamp can not be completely disabled.

The default behavior of the TCP/IP stack on this Systems is to not use the Timestamp options when initiating TCP connections, but use them if the TCP

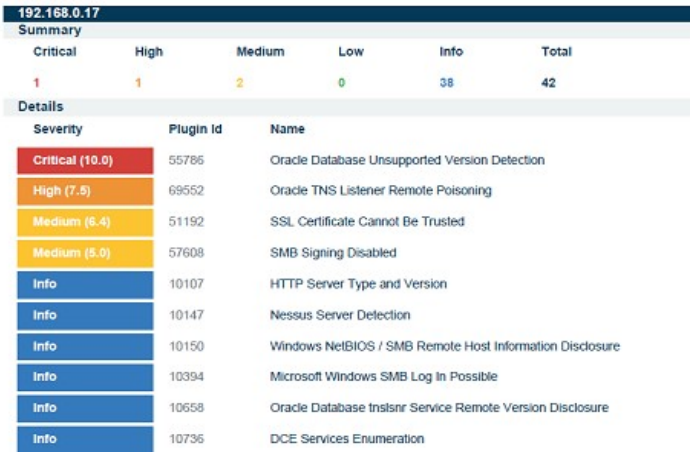
See also: http://www.microsoft.com/en-us/download/details.aspx?id=9152

Figura 2.19 Vulnerabilidad con Riesgo Bajo

- **Nessus**

  También utilizamos la herramienta Nessus para encontrar vulnerabilidades en los hosts. Se hizo empleo del Scanner tipo Advanced Scan para el host NEO69, y del tipo Web Application Tests para el host ALEX (Kali).

**Host NEO69 192.168.0.17**



192.168.0.17

Summary

| Critical | High | Medium | Low | Info | Total |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 0 | 38 | 42 |

Details

| Severity | Plugin Id | Name |
|---|---|---|
| Critical (10.0) | 55786 | Oracle Database Unsupported Version Detection |
| High (7.5) | 69552 | Oracle TNS Listener Remote Poisoning |
| Medium (6.4) | 51192 | SSL Certificate Cannot Be Trusted |
| Medium (5.0) | 57608 | SMB Signing Disabled |
| Info | 10107 | HTTP Server Type and Version |
| Info | 10147 | Nessus Server Detection |
| Info | 10150 | Windows NetBIOS / SMB Remote Host Information Disclosure |
| Info | 10394 | Microsoft Windows SMB Log In Possible |
| Info | 10658 | Oracle Database tnslsnr Service Remote Version Disclosure |
| Info | 10736 | DCE Services Enumeration |

Figura 2.20 Advanced Scan con Nessus

Como podemos observar en la figura 2.20, existe 1 vulnerabilidad crítica, 1 con riesgo alto y 2 con riesgo medio. También el reporte indica 38 Info, pero pasaremos por alto las vulnerabilidades de tipo Info.

En cuanto a la vulnerabilidad de riesgo medio (6.4) SSL Certificate Cannot Be Trusted, ella se basa en el certificado SSL propio de Nessus. Esto se origina ya que la versión que usamos para el análisis es el Free Trial. La solución es comprar la herramienta para obtener el certificado apropiado para este servicio.

**Host ALEX (Kali) 192.168.0.22**

Sólo se registraron vulnerabilidades de tipo Info, las cuales no tienen efecto en este análisis.

### 2.5.2 Análisis de Vulnerabilidades en servidores Web

Utilizamos la herramienta **Nikto** [5] para efectuar el análisis de los servidores Web instalados en el host NEO69 y host ALEX (Kali).

**Host NEO69 192.168.0.17**

La aplicación Web se encuentra alojada en el servidor Web WebLogic, el cual está instalado en este host.

Ejecutamos el siguiente comando de Nikto:

```
nikto -h http://192.168.0.17:7001
```


Figura 2.21 Comando nikto en Host NEO69

Como resultado de la ejecución, en la figura 2.21 observamos que no se registraron vulnerabilidades en el servidor Web del host NEO69.

**Host ALEX (Kali) 192.168.0.22**

En este host se encuentra instalado el BRMS y el directorio de seguridad OpenLDAP.

Ejecutamos el siguiente comando de Nikto para el análisis del servidor Web Apache-Coyote, el cual aloja la consola Web del BRMS:

```
nikto -h 192.168.0.22:8080
```


Figura 2.22 Comando nikto en consola Web del BRMS

Como resultado de la ejecución, en la figura 2.22 observamos que sí se encontraron 2 vulnerabilidades respecto a métodos HTTP:

- OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server.
- OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.

Cabe recalcar, que la vulnerabilidad OSVDB-397 encontrada en el host ALEX (Kali), está relacionada con la vulnerabilidad de Riesgo Alto CVE-2011-2900 encontrada en el host NEO69. La explotación exitosa permitirá a atacantes remotos ejecutar código arbitrario dentro del contexto de la aplicación afectada. Los intentos fallidos de explotación resultarán en una condición de denegación de servicio (Buffer overflow).

**Resultado del Análisis de Vulnerabilidades**

Tabla 3 Vulnerabilidades encontradas en hosts

| | **VULNERABILIDAD** | **RIESGO** | **HOST** |
|---|---|---|---|
| 1 | Mongoose Web Server Remote Buffer Overflow (Http PUT) | Alto | 192.168.0.17 |
| 2 | DCE/RPC and MSRPC Services | Medio | 192.168.0.17 |

| | | | |
|---|---|---|---|
| | Enumeration Reporting | | |
| 3 | TCP timestamps | Bajo | 192.168.0.17 |
| 4 | TCP timestamps | Bajo | 192.168.0.22 |
| 5 | OSVDB-397 Http Method PUT | Alto | 192.168.0.22 |
| 6 | OSVDB-5646 Http Method DELETE | Alto | 192.168.0.22 |
| 7 | Oracle Database Unsupported Version Detection | Crítico | 192.168.0.17 |
| 8 | Oracle TNS Listener Remote Poisoning | Alto | 192.168.0.17 |
| 9 | SMB Signing Disabled | Medio | 192.168.0.17 |

En el siguiente capítulo ejecutaremos los respectivos exploits en ambos hosts.

## 2.6 PENTESTING

En esta sección explotaremos las vulnerabilidades más importantes de las descritas en la tabla anterior, haciendo uso de varias herramientas y métodos.

- Vulnerabilidad 1: Mongoose Web Server Remote Buffer Overflow (Http PUT) en host 192.168.0.17.

  Esta vulnerabilidad tiene estrecha relación con Oracle WebLogic Server Apache Connector buffer overflow.

  Las siguientes acciones se realizaron con el firewall de Windows desactivado.

  Se efectuaron varios intentos en subir archivos con código malicioso (backdoor, DoS) al servidor WebLogic vía Http PUT requests, ninguno con éxito. Se utilizaron herramientas como Metasploit [9] (auxiliary/scanner/http/http_put - exploit/multi/handler), Weevely, MsfVenom, Poster, scripts con Python, Nmap, BurpSuite y Curl.

  El exploit con PUT requests resultó en un **falso positivo**, esto debido a la seguridad en los archivos de configuración de WebLogic y en el código de la aplicación, donde se utilizan

anotaciones, patrones de diseño seguros y librerías de seguridad con Java.

Se adjunta evidencia de ciertas acciones para el exploit:

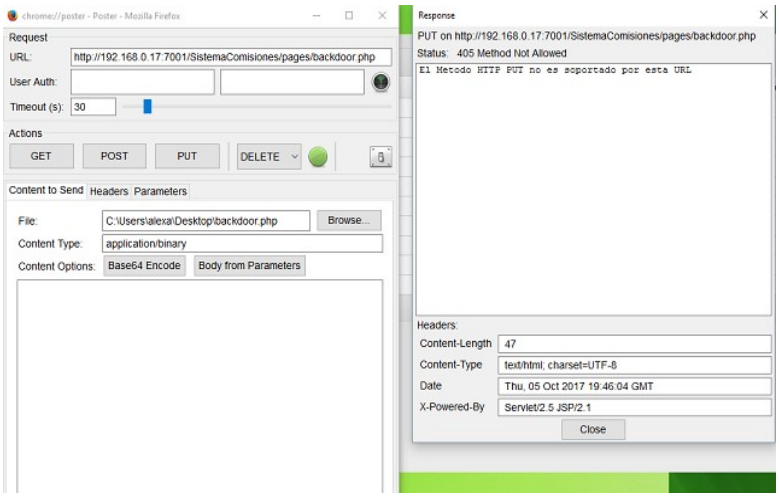



Figura 2.23 Poster


Figura 2.24 MSF Http_put (falso positivo)

```
import socket, sys

buffer = '\x41' * 3000

s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(('192.168.0.17',7001))

s.sendall(b'PUT /' + buffer.encode('utf-8') + b'/ HTTP/1.0\r\n')
s.sendall(b'\r\n')
print (s.recv(1024))
s.close()
print ('Exploit buffer sent successfully')
```

Figura 2.25 Script con Python DoS (falso positivo)

La siguiente acción se desarrolló con el firewall de Windows activo. También se realizó un ataque de DoS Http utilizando Slowloris [8] (script en Perl). Los ataques HTTP lentos son ataques de denegación de servicio (DoS) en los que el atacante envía las solicitudes HTTP en piezas lentamente, de a una por vez a un servidor web. Si una solicitud HTTP no está completa, o si la tasa de transferencia es muy baja, el servidor mantiene sus recursos ocupados esperando el resto de los datos. Cuando el connection pool del servidor alcanza su máximo, esto crea un DoS. Los ataques HTTP lentos son fáciles de ejecutar porque requieren recursos mínimos del atacante.

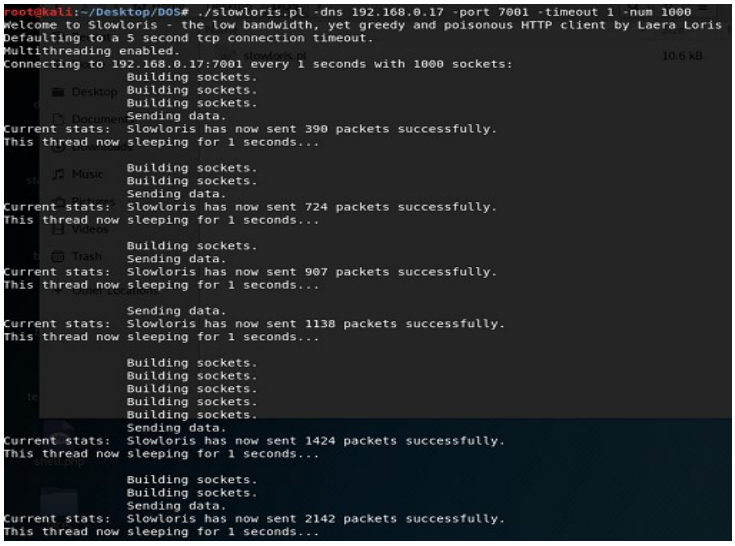



Figura 2.26 DoS con Slowloris.pl

Este ataque tuvo éxito ya que la aplicación Web de Comisiones dejó de responder, como observamos en la siguiente figura.

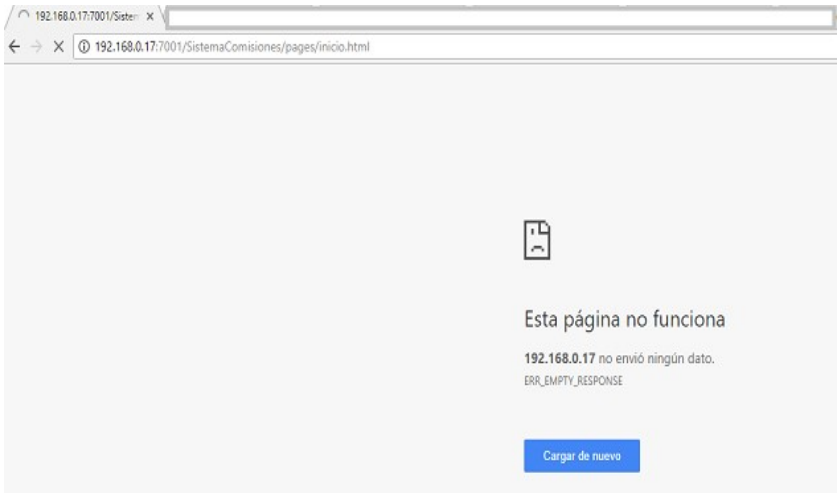



Figura 2.27 Aplicación Web afectada con Slowloris

Cuando se detuvo el script, la aplicación Web respondió correctamente.

- Vulnerabilidades 5 y 6: OSVDB-397 Http Method PUT y OSVDB-5646 Http Method DELETE en host 192.168.0.22.

Las siguientes acciones se realizaron con el firewall UFW disabled.

Como primera acción se ganó acceso al host .22, utilizamos la herramienta Hydra [11] para obtener las credenciales, el cual es un cracker de inicio de sesión paralelizado que admite numerosos protocolos para atacar. Es muy rápido y flexible. Esta herramienta permite a los investigadores y consultores de seguridad mostrar cuán fácil sería obtener acceso no autorizado a un sistema de forma remota.

```
hydra -l root -P /usr/share/dirb/wordlists/small.txt -t 6 ssh://192.168.0.22
```

Luego de obtener las credenciales con usuario root, nos conectamos vía SSH al host, y nos dirigimos al directorio donde está instalado el JBoss Server. Indagando en el directorio del

server y BRMS, podemos encontrar los archivos .properties donde están las credenciales que necesitamos de la consola Web y WebDav.

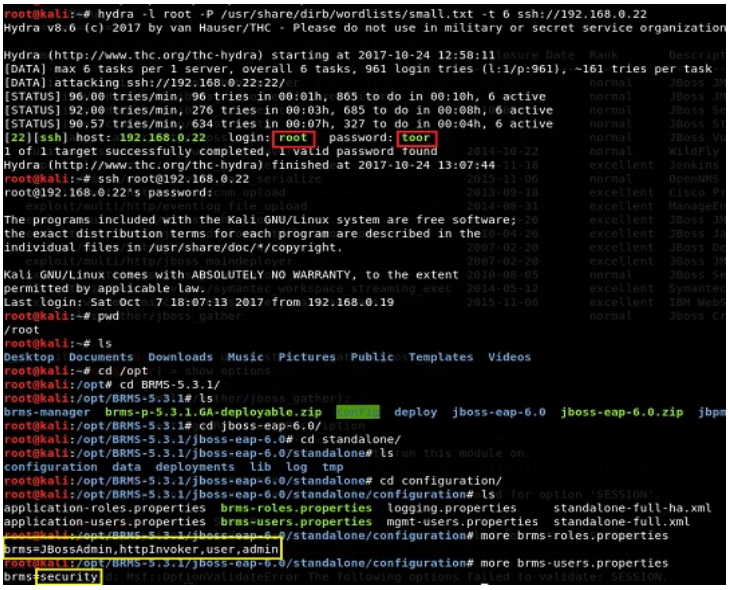



Figura 2.28 Credenciales obtenidas del BRMS

El repositorio del BRMS se puede acceder también a través de WebDav. WebDAV es una interfaz del sistema de archivos basada en HTTP que puede ser accedida desde cualquier lugar. Los paquetes que contienen las reglas del negocio pueden ser consultados o explotados vía WebDav.

La URL para acceder al repositorio utilizando WebDav:

http://192.168.0.22:8080/jboss-brms/org.drools.guvnor.Guvnor/webdav/

Autenticación es requerida, pero ya obtuvimos las credenciales.

Cadaver es una herramienta en Kali de línea de comandos que admite la carga y descarga de archivos vía WebDav.

Haciendo uso de Cadaver vamos a explotar la vulnerabilidad del Http Put (si el método está habilitado, puede ser utilizado para atacar la aplicación mediante un archivo).

```
cadaver http://192.168.0.22:8080/jboss-brms/org.drools.guvnor.Guvnor/webdav/
```

Ahora estamos dentro del directorio de la víctima, con el comando put subimos los archivos shell.php y prueba.dlr (la extensión DRL viene de Drools Rule Language y es un lenguaje de programación para definir reglas de negocio) en el directorio:

.. /packages/ahorro.ADMINISTRATIVA/

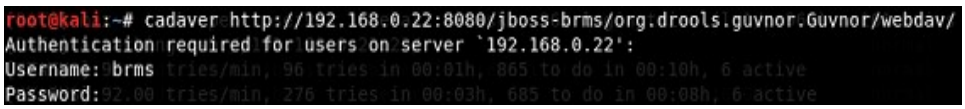



Figura 2.29 Obtener acceso vía Cadaver

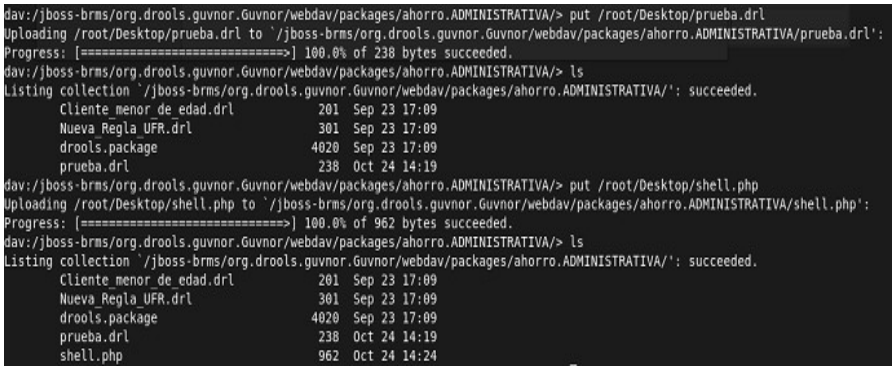



Figura 2.30 Archivos subidos al servidor

```
salience 4
dialect "java"
activation-group "comision"
when
        m:Comision( ( EdadClie <= 18) )
then
        m.setTarifa(500.0);
        m.setMoneda_regla('CLP');
        m.setRegla(kcontext.getRule().getName());
        retract(m);
```

Figura 2.31 Regla de negocio con lenguaje DLR

Con esta explotación demostramos que es factible subir archivos al servidor, como ejemplos un backdoor o archivo. dlr con reglas de negocio fraudulentas (creación y modificación).

Poster es una herramienta de desarrollo para interactuar con servicios web y otros recursos web que le permite realizar solicitudes HTTP, establecer el cuerpo de la entidad y el tipo de contenido.

A continuación, utilizando Poster y Cadaver vamos a explotar la vulnerabilidad del Http Delete vía WebDav. Los archivos que subimos en la explotación anterior vamos a eliminar, shell.php y prueba.dlr.

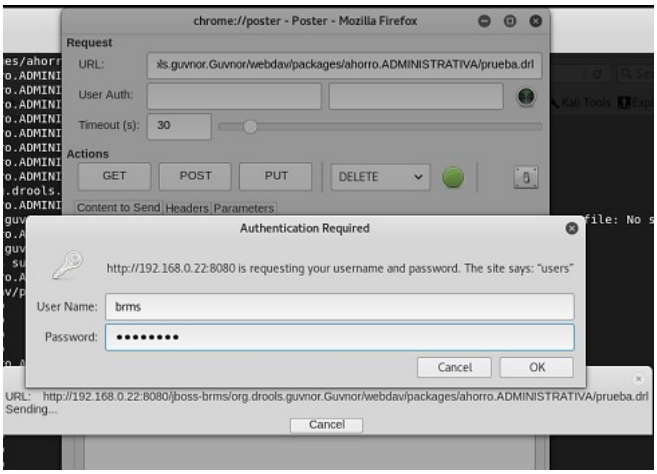



Figura 2.32 HTTP Delete con Poster

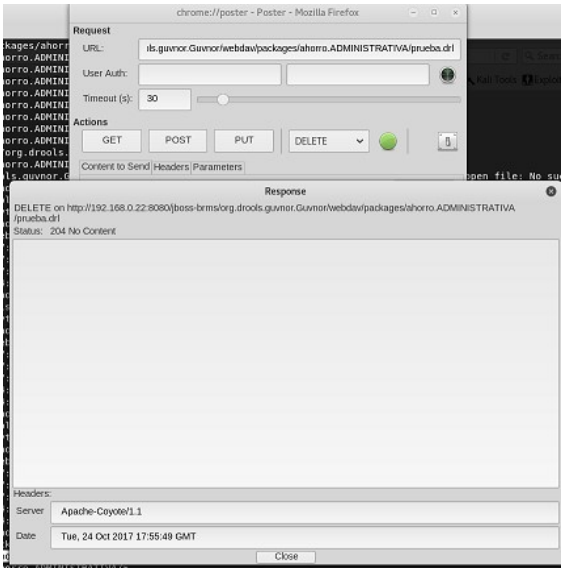



Figura 2.33 Confirmación de Poster

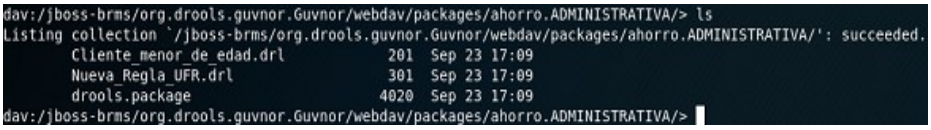



Figura 2.34 Package actualizado

Con Cadaver también podemos eliminar archivos haciendo uso del comando **rm.**

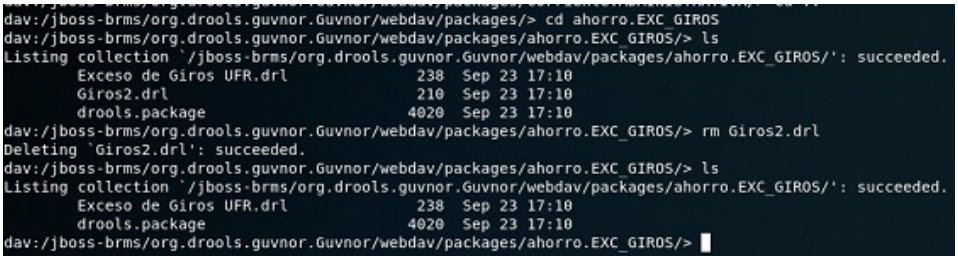



Figura 2.35 Comando rm y Cadaver

También se realizó un ataque de DoS (Syn Flood) utilizando Hping3 [7]. Hping es una herramienta en línea de comandos que nos permite crear y analizar paquetes TCP/IP, y como tal tiene muchas utilidades: hacer testing de firewalls, escaneo de puertos, redes y también tiene la capacidad de provocar un SYN Flood Attack mediante denegación de servicio (DoS). El objetivo de un ataque de este tipo es el envío de peticiones de conexión TCP más rápido de lo que una máquina puede procesar, a fin de saturar los recursos y evitar que la máquina pueda aceptar más conexiones.

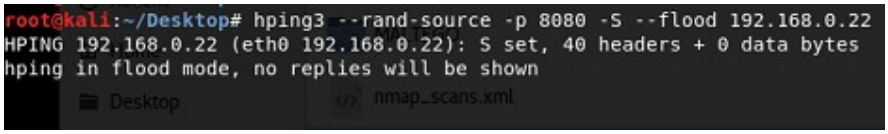



Figura 2.36 Syn Flood con hping3

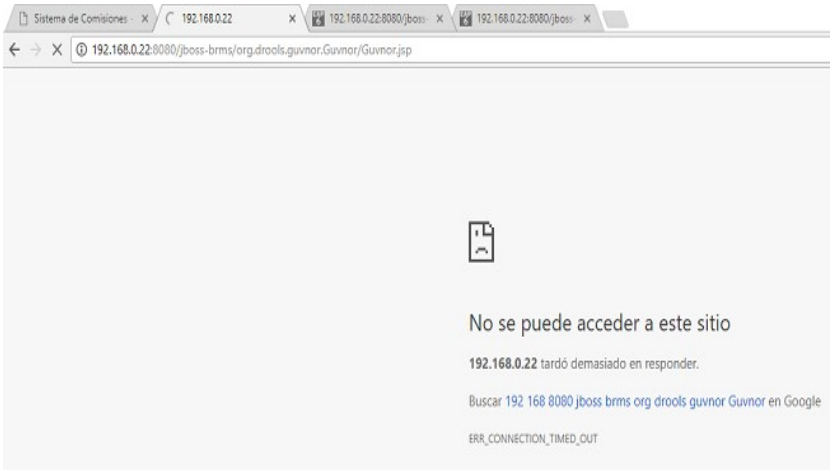



Figura 2.37 Consola Web del BRMS caída

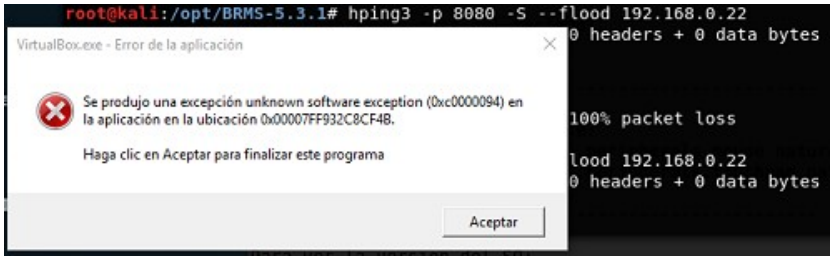



Figura 2.38 Excepción en OVB

Como podemos apreciar en las imágenes anteriores, la ejecución del comando hping3 resultó en la caída del servidor JBoss, OVB, y en una

ocasión la máquina física hacker con Windows 10 nos entregó el pantallazo azul.

**Resultados de Exploits**

Tabla 4 Exploits ejecutados

| ACCIÓN | ESTADO | HOST |
|---|---|---|
| Http Put a servidor WebLogic | Falso Positivo | 192.168.0.17 |
| DoS con Slowloris a WebLogic | Exitoso | 192.168.0.17 |
| Obtener credenciales del host con Hydra | Exitoso | 192.168.0.22 |
| Acceso SSH al host | Exitoso | 192.168.0.22 |
| Obtener credenciales del BRMS | Exitoso | 192.168.0.22 |
| Http Put con Cadaver a JBoss | Exitoso | 192.168.0.22 |
| Http Delete con Cadaver a JBoss | Exitoso | 192.168.0.22 |
| Http Delete con Poster a JBoss | Exitoso | 192.168.0.22 |
| DoS con Hping3 a JBoss | Exitoso | 192.168.0.22 |

## 2.7 IMPLEMENTACIÓN DE OWASP SONAR QUBE

SonarQube es una de las herramientas (software libre) más populares del mundo, permite gestionar la calidad del código fuente y es utilizada activamente por muchos desarrolladores y compañías.

Al instalarla podremos recopilar, analizar, y visualizar métricas del código fuente. También realiza un histórico de todas las métricas del proyecto y permite visualizar informes con resúmenes de las métricas.

Trabaja principalmente para Java, aunque existe soporte para varios lenguajes de programación.

El primer paso para realizar el análisis del código fuente con SonarQube es iniciar el servidor. Luego, en el dir de Maven, buscamos el archivo de configuración settings.xml y agregamos los siguientes tags:

```
<pluginGroups>
        <pluginGroup>org.sonarsource.scanner.maven</pluginGroup>
</pluginGroups>

<profile>
        <id>sonar</id>
        <activation>
          <activeByDefault>true</activeByDefault>
        </activation>
        <properties>
          <sonar.host.url>
            http://localhost:9000
          </sonar.host.url>
        </properties>
</profile>
```

En el directorio donde está ubicado el fuente, al nivel del archivo pom.xml, creamos el archivo sonar-project.properties con el siguiente contenido:

```
sonar.projectKey=SistemaComisiones
sonar.projectName=SistemaComisiones
sonar.projectVersion=1.0
sonar.sources=/
```

Ya se encuentra lista la configuración para el análisis.

Ahora mediante la línea de comandos, nos situamos en el dir del fuente y con el comando **mvn sonar:sonar** se inicia el scanning.


Figura 2.39 Scanning con SonarQube

…

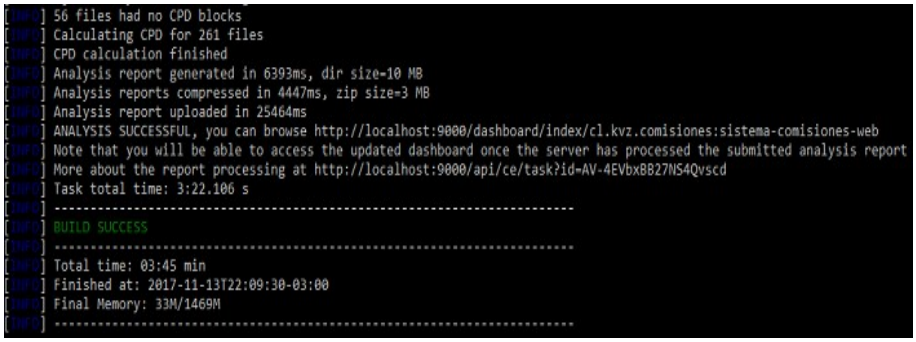



Figura 2.40 Build Success

Mediante la URL http://localhost:9000 tenemos acceso a los proyectos y métricas. Se aprecia en la figura 2.41, que la calidad del código fuente fue **Aprobada (Passed).**

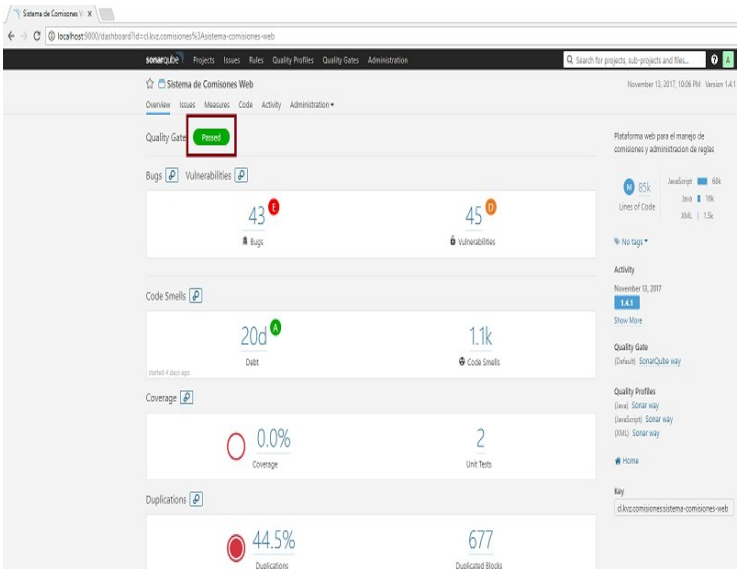



Figura 2.41 Código fuente Aprobado

El reporte indica que fueron encontrados 43 bugs y 45 vulnerabilidades, pero cabe destacar, la gran mayoría de ellos no son críticos y pertenecen a AngularJS, un framework de JavaScript de código abierto utilizado en la aplicación Web.

# CAPÍTULO 3

# ANÁLISIS DE RESULTADOS

## 3.1 ANÁLISIS DE RESULTADOS DE EXPLOITS

- Host .17: El ataque Http Put al servidor WebLogic resultó en un falso positivo, mientras que el ataque DoS con Slowloris.pl tuvo éxito ya que la aplicación Web de Comisiones dejó de responder.
- Host .22: Se obtuvo credenciales del host con Hydra, luego vía SSH logramos acceso y encontramos en el directorio del Jboss Server las credenciales del BRMS.

  Con Cadaver se pudo explotar los métodos Put / Delete, y con Poster el método Delete. Mediante Hping3 tuvimos éxito en la

caída del JBoss Server, OVB, e inclusive en una ocasión la máquina física hacker con Windows 10.

## 3.2 ACCIONES PREVENTIVAS Y CORRECTIVAS

A continuación, se detallarán las acciones de prevención y/o corrección por cada vulnerabilidad encontrada y exploits ejecutados en hosts y servidores Web.

Se investigaron y probaron varios tipos de mitigaciones; así también estarán descritas ciertas soluciones provistas por los analizadores de vulnerabilidades utilizados.

- **DoS con Slowloris a servidor WebLogic en host .17**

  Para reducir la posibilidad de ataques de denegación de servicio (DoS), Oracle recomienda que configure el parámetro “message timeout” para el servidor [10]. De forma predeterminada, el servidor espera 60 segundos para recibir el mensaje completo. La duración del tiempo de espera se establece en un nivel alto para acomodar conexiones lentas. Se debe reducir el parámetro “message timeout” a la configuración más baja posible.

Otra práctica recomendada para evitar ataques DoS es restringir el tamaño del mensaje (el valor predeterminado es 10 MB). También se puede limitar la cantidad de sockets permitidos para un servidor configurando la opción “Maximun Open Sockets” en la página de configuración del servidor.

- **Obtener credenciales del host .22 con Hydra**

  Se recomienda confeccionar contraseñas robustas y extensas, optar siempre por combinaciones alfanuméricas, intercalar signos de teclado, cambiarlas frecuentemente, y así podemos evitar ataques de fuerza bruta, de diccionario, ingeniería social, etc.

- **Acceso SSH al host .22**

  Una buena táctica de seguridad para el protocolo SSH es cambiar el número de puerto 22 (valor por default) para evitar posibles ataques. También se recomienda deshabilitar el usuario root para login y sólo habilitar IPs designadas con reglas de firewall.

  Para lograr esto, editamos el archivo **/etc/ssh/sshd_config**

  Port 7720

  PermitRootLogin no

  AuthorizedKeysFile .ssh/authorized_keys

También creamos un nuevo usuario con privilegios de administrador, así evitamos utilizar el usuario root en conexiones remotas.

```
#useradd -m username
#passwd username
#usermod -a -G sudo username
#chsh -s /bin/bash username
```

- **Http Put con Cadaver a servidor JBoss en host .22**

  Se agregó la siguiente regla al firewall UFW:

```
iptables -I INPUT -d 192.168.0.22 -p tcp --dport 8080 -m string --
string 'PUT /' --algo bm -j DROP
```

  Con esta regla se filtró el http put request.

- **Http Delete con Cadaver / Poster a servidor JBoss en host .22**

  Se agregó la siguiente regla al firewall UFW:

```
iptables -I INPUT -d 192.168.0.22 -p tcp --dport 8080 -m string --
string 'DELETE /' --algo bm -j DROP
```

  Con esta regla se filtró el http delete request.

- **DoS con Hping3 a servidor JBoss en host .22**

  Las políticas adecuadas de filtrado en firewalls suelen ser la primera línea de defensa contra DoS, sin embargo, el kernel de Linux también puede reforzarse contra este tipo de ataques. Este tipo de hardening es útil para las SYN floods [2] que intentan sobrecargar un servicio en particular con solicitudes (como http) en oposición a una que intenta saturar la conexión de red del servidor, para lo cual se necesita un firewall para evitarlo.

  El kernel de Linux permite cambiar directamente los diversos parámetros necesarios para mitigar los ataques de SYN flood. Primero, estableceremos que las variables estén activas de forma inmediata:

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
echo 2048 > /proc/sys/net/ipv4/tcp_max_syn_backlog
echo 3 > /proc/sys/net/ipv4/tcp_synack_retries
```

Esto establece que el kernel use el mecanismo de cookies SYN, use un tamaño de cola de backlog de 2048 conexiones y la cantidad de tiempo para mantener las conexiones semiabiertas en la cola (3 equivale aproximadamente a 45 segundos).

Para que estos cambios persistan durante los reinicios consecutivos, debemos informar al sistema sysctl acerca de estos parámetros modificados. Usamos el archivo /etc/sysctl.conf para hacerlo. Agregaremos las siguientes líneas al final del archivo:

```
# TCP SYN Flood Protection
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 2048
net.ipv4.tcp_synack_retries = 3
```

Los cambios ahora serán permanentes.

También se recomienda el uso de firewalls como PFSense y Sophos para prevenir ataques Syn flood.

- **Vulnerabilidad DCE/RPC and MSRPC Services Enumeration Reporting en host .17**

  Solución: Mitigación. Es necesario filtrar el tráfico entrante a los puertos indicados por OpenVas.

- **Vulnerabilidad TCP timestamps en hosts .17 y .22**

  Para desactivar las marcas de tiempo de TCP en Windows, ejecute

  ```
  netsh int tcp set global timestamps = disabled
  ```

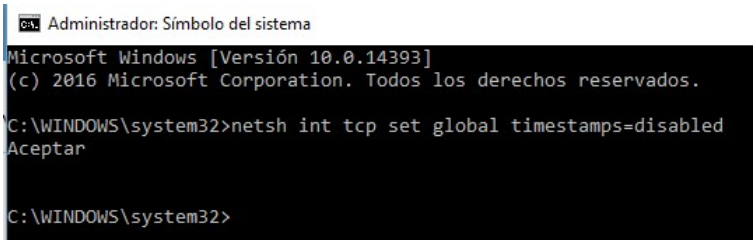



Figura 3.1 Netsh en Windows

- **Vulnerabilidad Oracle Database Unsupported Version Detection en host .17**

  Se utilizó en ambiente de desarrollo un listener del cliente Oracle para 32 bits (versión 10.2.0.5). Es necesario actualizar el cliente Oracle a la versión 11 o 12 para 64 bits.

- **Vulnerabilidad Oracle TNS Listener Remote Poisoning en host .17**

  Ir al directorio:

  C:\oraclexe\app\oracle\product\10.2.0\server\NETWORK\ADMIN

  Abrir LISTENER.ora y agregar la siguiente línea al archivo:

  DYNAMIC_REGISTRATION_LISTENER = OFF

  Luego reiniciar el listener.

- **Vulnerabilidad SMB Signing Disabled en host .17**

SMB Signing se usa para garantizar que los paquetes SMB no se modifiquen durante el tránsito (ej.: man in the middle attack). De forma predeterminada, todos los servidores, clientes y centros de datos de Windows admiten SMB Signing, pero no están habilitadas. Un cliente no podrá establecer una sesión con un servidor que tenga habilitada SMB Signing hasta que el cliente acepte firmar SMB. SMB Signing puede provocar una degradación del rendimiento de hasta un 15% o superior, ya que hay una sobrecarga con el servicio.

SMB Signing se puede habilitar a través del regedit:

En estaciones Windows:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanManWorkstation\Parameters

Editar los siguientes registros:

EnableSecuritySignature – Value info = 1 (enable)

RequireSecuritySignature – Value info = 1 (enable)

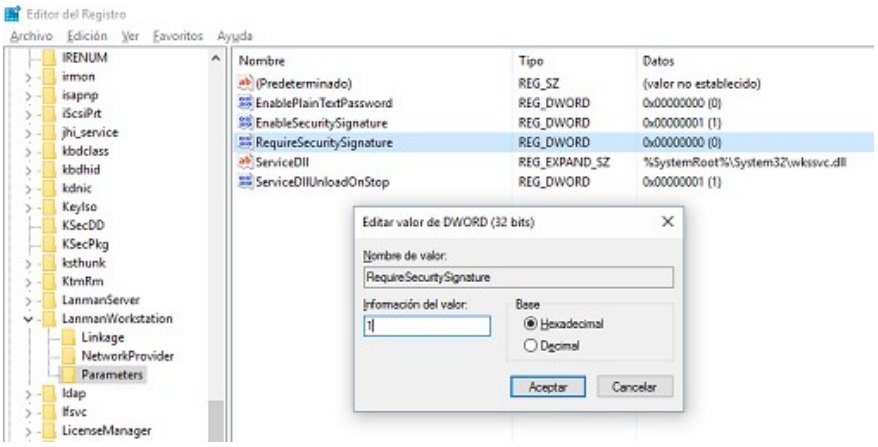



Figura 3.2 Editor de Registro

# CONCLUSIONES

1. Oracle WebLogic Server es la plataforma Java EE con la mayor población, productividad y desempeño. El presente trabajo se realizó en un ambiente de desarrollo, por lo tanto, la configuración de seguridad del servidor se encuentra parcialmente activa. En producción, la seguridad debe ser ampliamente configurada.

2. Se utilizaron varias herramientas para el ataque Http Put al servidor WebLogic (host NEO69), pero resultó en un falso positivo ya que WebLogic implementa el tipo de seguridad Http requests en sus archivos de configuración. También el código de la aplicación Web presenta restricciones mediante anotaciones y clases en Java.
Sin embargo, también se realizó un ataque de DoS Http utilizando Slowloris.pl con el firewall de Windows activo. El exploit tuvo éxito ya que la aplicación Web dejó de responder.

3. Se logró obtener las credenciales del host ALEX (Kali) y vía SSH las credenciales del BRMS. Se explotó los Http requests Put y Delete en el servidor JBoss. Y también mediante Hping3 se efectuó la caída del servidor, OVB, e inclusive en una ocasión la máquina física hacker con Windows 10.

4. Ambos servidores web se encuentran vulnerables a exploits de Denegación de servicio. Y el servidor JBoss a exploits Put y Delete.

5. Analizando el código fuente de la aplicación Web del Sistema de Comisiones con SonarQube, podemos determinar que la calidad del código fue Aprobada (Passed).

# RECOMENDACIONES

1. Emplear una mayor capacidad y ancho de banda combinado con la redundancia del servicio puede reducir la susceptibilidad a algunos ataques DoS. El servidor de aplicaciones debe poder configurarse para soportar o minimizar el riesgo de ataques DoS. Esto se puede lograr parcialmente si el servidor de aplicaciones proporciona opciones de configuración que limiten el número permitido de conexiones HTTP concurrentes. Se puede establecer tres atributos en el servidor WebLogic que ayudan a prevenir este tipo de ataque:

PostTimeoutSecs

MaxPostTimeSecs

MaxPostSize

2. Implementar SSL/TLS Deployment (HTTPS) en servidor WebLogic.

3. Iptables no es la mejor herramienta para el filtrado HTTP. Se recomienda el uso de un HTTP proxy como Nginx o SynProxy para tener un control más flexible y óptimo sobre todas las solicitudes HTTP que se reciban.

4. Hacer empleo de algún firewall WAF (Ej: OWASP ModSecurity) con reglas correctas, IPS / IDS, antivirus actualizado, software de seguridad y hardening completo en servidores [12].

5. Establecer contraseñas robustas.

6. Es necesario actualizar el cliente Oracle a la versión 11 o 12 para 64 bits [13].

# BIBLIOGRAFÍA


[1] Karina Astudillo, Hacking Ético 101. Cómo hackear profesionalmente en 21 días o menos, 2013.

[2] https://www.ndchost.com/wiki/server-administration/hardening-tcpip-syn-flood

[3] https://enekoamieva.com/instalar-openldap-en-servidor/

[4] http://www.hackingtutorials.org/scanning-tutorials/installing-openvas-kali-linux/

[5] https://thehackerway.com/2011/05/12/conceptos-basicos-de-nikto-tecnicas-de-escaneo-de-servidores-y-aplicaciones-web/

[6] http://www.drchaos.com/enable-ssh-on-kali-linux-enable-ssh-on-kali-linux/

[7] http://www.draconyx.net/articles/firewall-testing-using-hping3.html

[8] https://drive.google.com/file/d/0ByEMZENvukL6TkN5ak1GTmFQc2c/view

[9] https://www.rapid7.com/db/modules/auxiliary/scanner/http/http_put

[10] https://docs.oracle.com/cd/E13222_01/wls/docs61/security/lockdown.html

[11] https://null-byte.wonderhowto.com/how-to/hack-like-pro-crack-online-web-form-passwords-with-thc-hydra-burp-suite-0160643/

[12] https://www.pluralsight.com/blog/it-ops/linux-hardening-secure-server-checklist

[13] http://www.oracle.com/technetwork/topics/winx64soft-089540.html

[14] https://www.sonarqube.org/