Màster Interuniversitari en Seguretat de les TIC (MISTIC)

TFM: Honeypot

Autor: **Isaac Yera Caballero**

Director: **Carles Estorach Espinós**

Data: 23/06/2014

Contacte: iyeraj@uoc.edu

Autor: Isaac Yera Caballero
Contacte: iyera@uoc.edu

Dedicatòria i Agraïments

Per a tu Esther que sempre ets on i quan et necessito, sense tu res d'això seria possible.

Autor: Isaac Yera Caballero
Contacte: iyera@uoc.edu
Data: 23/06/14

Resum del projecte

Català:

L'objectiu del projecte consisteix en la implementació de un honeypot que emula tecnologies basades en entorns Microsoft i Linux. Per, posteriorment rebre atacs i extreure informació sobre la tipologia dels mateixos i la metodologia feta servir pels atacants per portar-los a fi.

Un cop extreta tota la informació dels atacs es confeccionarà un informe detallat que reflectirà qui, com i què es atacat.

English:

The objective of this project is build a honeypot which emulates a Microsoft Windows and Linux environments for take in attacks. Later, we extract information from attacks to get their type and methodology used by aggressors.

With the attack information we will make a report with the most relevant information, which detail reveals the most important information, Who, How and Which is attacked.

Índex

Índex de figures

Autor: Isaac Yera Caballero
Contacte: iyera@uoc.edu

1. Introducció

Un Honeypot és un software o conjunt de computadores que simulen ser dèbils i vulnerables per tal d'atreure atacants, recol·lectar informació sobre atacants i tècniques utilitzades i per últim desviar l'atenció de les màquines importants del sistema.

Aquest projecte te propòsit d'investigar el modus operandum que té un hacker al moment en que intenta assaltar un sistema amb o sense èxit. Amb aquesta finalitat es faran servir Honeypots amb tecnologies Linux i Windows per poder veure com varien els atacs i les seves conseqüències en funció de la tecnologia utilitzada.

Per fer possible l'anàlisi de la conducta dels atacants es muntarà un entorn Honeypot amb dades fictícies amb les quals volem atreure als atacants per tal de monitoritzar la seva activitat i així poder extreure patrons.

2. Objectius

L'objectiu d'aquest projecte es muntar un entorn Honeypot per atreure atacants, aconseguir el seu patró/vector d'atac en funció de la tecnologia atacada. Per posteriorment fer un petit estudi sobre:

- Quin és el sistema o servei atacat?
- Qui ataca el sistema o servei? I quina és la procedència de l'atac?
- Quines accions realitzen per fer-se amb el control del sistema o servei?
- Com s'aprofiten del sistemes o serveis compromesos?

Més concretament la llista d'objectius a assolir pel projecte són:

- El disseny d'una arquitectura de xarxes i sistemes que facilitin l'entrada d'atacants i la recol·lecció de informació.
- Selecció d'eines de monitorització que facilitin la captació de dades i esdeveniments generats pels atacants.
- Implementar els sistemes Honeypot i les eines de monitorització seleccionats.
- Estudi dels atacs rebuts en funció de les dades recol·lectades.

Als objectius del propis del projecte cal afegir els objectius personals que en el meu cas ha sigut la possibilitat d'aprofundir en l'interessant món dels Honeypots i sobretot en aprendre quins són els interessos i la metodologia feta servir pels atacants.

3. Planificació

3.1. Tasques

Per tal d'assolir els objectius descrits es realitzaran les següents tasques:

Tasca	Descripció
Estudi de viabilitat	- Definir els requisits del projecte - Estudiar les diferents alternatives - Valoració de les alternatives i selecció de la solució
Anàlisi i disseny del sistema	- Definir detalladament cada part del sistema - Definir l'arquitectura per afavorir els atacs i la posterior recol·lecció de dades.
Implantació del projecte	- Estudiar i implementar els components seleccionats. - Realització de proves unitàries a cada part del sistema i del sistema en conjunt.
Període d'escolta	- Durant un període encara per determinar es deixarà el honeypot rebent atacs fins a aconseguir un conjunt de dades representatiu.
Recol·lecció de dades	- Després del període d'escolta es recol·lectaran i processaran les dades captades per tal d'analitzar els resultats
Redacció del informe	- Amb les dades recol·lectades al pas anterior es realitzarà un informe executiu que ressaltarà les parts més importants així com les conclusions obtingudes.

Fig 1: Taula de tasques

3.2. Planificació Temporal

Nombre de tarea	Duración	Comienzo	Fin
Estudi de viabilitat	15 días	lun 10/03/14	vie 28/03/14
Anàlisi i disseny del sistema	15 días	lun 31/03/14	vie 18/04/14
Implantació del projecte	10 días	lun 21/04/14	vie 02/05/14
Període d'escolta	15 días	lun 05/05/14	vie 23/05/14
Recol·lecció de dades	12 días	lun 26/05/14	mar 10/06/14
Redacció del informe	12 días	mié 11/06/14	jue 26/06/14
Lliurament PAC1	0 días	vie 14/03/14	vie 14/03/14
Lliurament PAC2	0 días	lun 14/04/14	lun 14/04/14
Lliurament PAC3	0 días	vie 23/05/14	vie 23/05/14
Lliurament Final	0 días	jue 26/06/14	jue 26/06/14

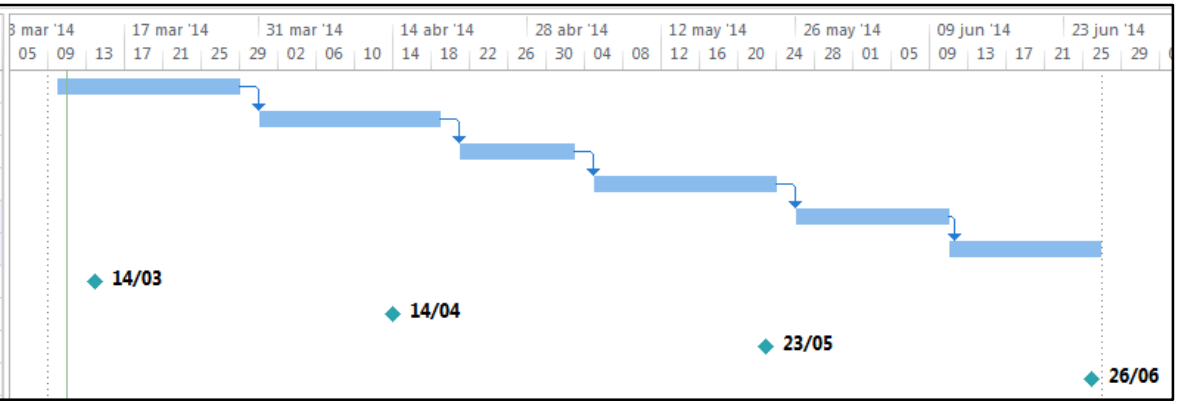

Fig 2: Taula amb planificació temporal

4. Estat de l'art

En aquest apartat es detallarà que són i com funcionen els Honeypots

4.1. Què és un Honeypot?

En conseqüència a la gran quantitat d'atacs que reben des de Internet les institucions públiques i privades s'han hagut de incrementar les mesures de seguretat. Una d'aquestes mesures ha sigut implementar sistemes trampa per a observar el comportament d'un atac i així poder analitzar la intrusió i la manera en que s'ha fet.

Els Honeypots son sistemes que simulen ser vulnerables i per tant perceptibles de ser atacats. Són molt útils per aprendre nous mètodes d'intrusió, capturar malware, desviar l'atenció dels atacants, etc.

Els Honeypots es poden classificar en varis grups segons la nivell d'interacció que un atacant pugui tenir amb ells:

- Baixa Interacció: Són els més fàcils d'implementar, tenen funcionalitats reduïdes i funcionen emulant serveis que podran ser atacats sense afectar a la resta del sistema en que està allotjat.
- Mitjana Interacció: Als de mitjana interacció els atacants tenen més llibertat per interactuar, fent que el honeypot sigui capaç d'emular el comportament d'un software.
- Alta interacció: Aquest últim tipus de Honeypot són els més costosos d'implementar i els que més informació ens donen sobre els atacants. Són més difícils d'implementar per que no s'emula cap mena de soft sinó que s'interactua amb tot el sistema operatiu.

4.2. On s'han d'ubicar les Honeypots?

Tenim varies opcions a l'hora d'ubicar el honeypot dins de la nostre xarxa, tot dependrà de l'arquitectura de la mateixa (segmentació, DMZ, ...). En el nostre cas destacarem dos exemples:

1. Sense DMZ

En el cas de no disposar de zona desmilitaritzada (DMZ) podem situar el nostre honeypot en dues posicions diferents: davant o darrere del Firewall.

Si posem el Honeypot davant el Firewall (entre Internet i el Firewall) evitem afegir riscos a la nostre xarxa interna i evitem falsos positius de sistemes IDS però podem incrementar notablement la transferència de xarxa perquè el honeypot està totalment desprotegit.

En el cas contrari, si posem el honeypot darrera del Firewall (entre el Firewall i la nostre xarxa interna) aquest quedarà afectat per les regles de filtrat del Firewall i harem de modificar-les per permetre l'accés i la part més perillosa és que potencialment, un atacant pot fer-se amb el control del honeypot i accedir d'aquesta manera a la xarxa interna.

2. Amb DMZ

Situar el honeypot a al DMZ té dos aspectes molt positius, per una banda permet situar-ho junt amb altres servidors i a l'hora evitem exposar la xarxa interna ja que existeix un Firewall que en protegeix.

5. Estudi de la viabilitat

En aquest capítol s'analitzaran els requisits del projecte, es proposaran alternatives per donar-li solució i per últim es valorarà quina alternativa s'adapta més als objectius definits pels requisits del projecte.

5.1. Definir requisits

L'objectiu principal d'aquest treball és estudiar quins són els vectors d'atac que reben els sistemes operatius. Com a conseqüència directa del objectius es definiran els requisits en funció dels objectius.

Per tal de portar a fi una anàlisi dels atacs que reben els sistemes operatius necessitarem crear un entorn connectat a Internet de forma directa que afavoreixi els atacs. I que a més sigui multiplataforma per analitzar els atacs en funció del sistema operatiu objectiu. Per tant, la solució ha de implementar-se de forma que pugui interceptar atacs que tinguin com objectiu sistemes Linux i Windows.

Per rebre els atacs de forma controlada i seguint les indicacions de l'enunciat del projecte, es farà servir Honeydrive que és una distribució Linux que ens permet crear honeypots de forma fàcil i controlada, a l'hora que incorpora eines per fer un anàlisis dels atacs.

5.2. Estudi i selecció d'alternatives

Com s'ha explicat en capítols anteriors, hi ha varis tipus de honeypots i es poden situar en diferents parts de la xarxa segons els nostres objectius.

5.2.1. Tipus de Honeypots

Segons la interacció que permet el honeypot aquests es poden classificar en tres tipus baixa, mitja i alta iteració.

Els honeypots de baixa iteració són els més fàcils d'implementar perquè gairebé només simula un port obert. Per exemple si volem imitar un MySQL només simularia obrir el port 3306 i com a molt respondre a les peticions de login de la mateixa manera en que ho fa el servei de MySQL.

Els honeypots de mitja iteració donen més llibertat d'actuació fent possible l'emulació parcial o total del software sel·leccionat. Si extrapolem l'exemple anterior a aquest nou escenari, a part d'emular el port podríem arribar a entrar al gestor de BDs i arribar a executar algunes queries.

Per últim tenim els honeypots d'alta interacció, són els més costosos d'implementar i bàsicament consisteix en agafar un sistema complert i exposar-ho als atacants per posteriorment analitzar els atacs que s'han realitzat.

Donat els requisits del nostre projecte i els temps amb que s'han planificat la implementació, execució i anàlisis dels atacs, el tipus de honeypot més adequat pel nostre escenari és el de mitja interacció. Degut a que guarda el perfecte equilibri entre dificultat, temps de posada en funcionament, i les dades que obtindrem de la anàlisis dels atacs rebuts. La implementació dels honeypots es realitzarà sobre la distribució Honeydrive que es detallarà més endavant.

5.2.2. Ubicació del Honeypot

En un entorn empresarial, molt possiblement tindríem ubicat el honeypot en les posicions indicades en capítols anteriors:

- En el cas de no tenir DMZ podria estar davant o darrera del Firewall

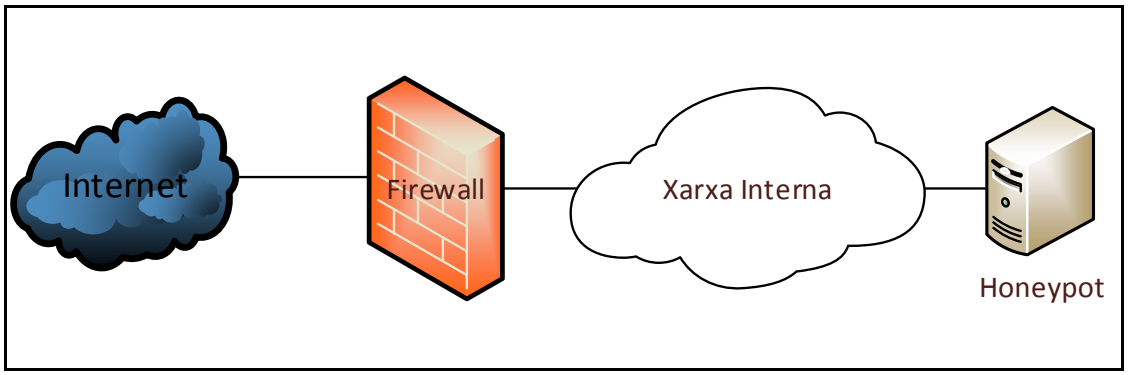

Fig 3: Topologia de xarxa sense DMZ, honeypot darrera del firewall

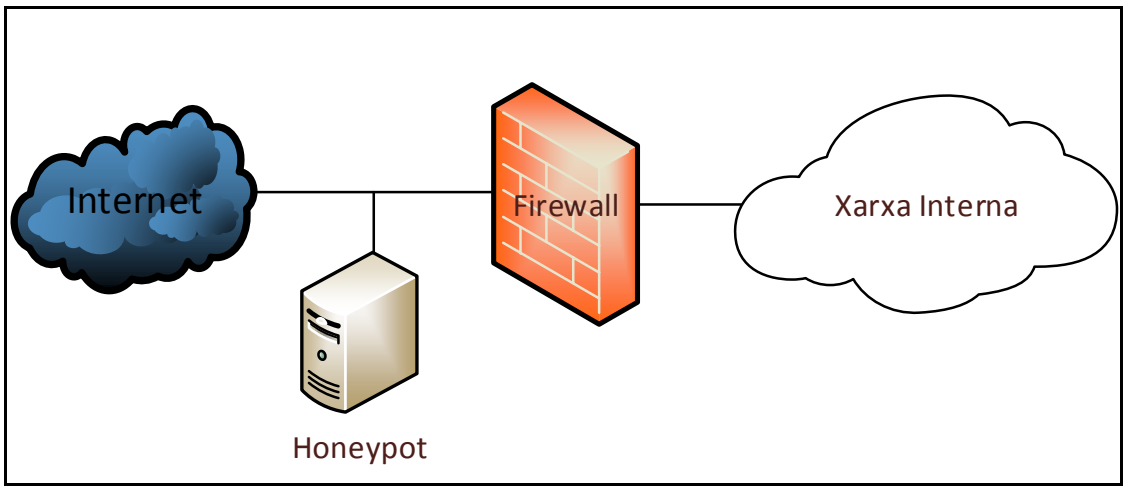

Fig 4: Topologia de xarxa sense DMZ, honeypot davant del firewall

- En cas de tenir DMZ es pot situar en aquesta junt amb la resta de servers exposats de l'entitat.

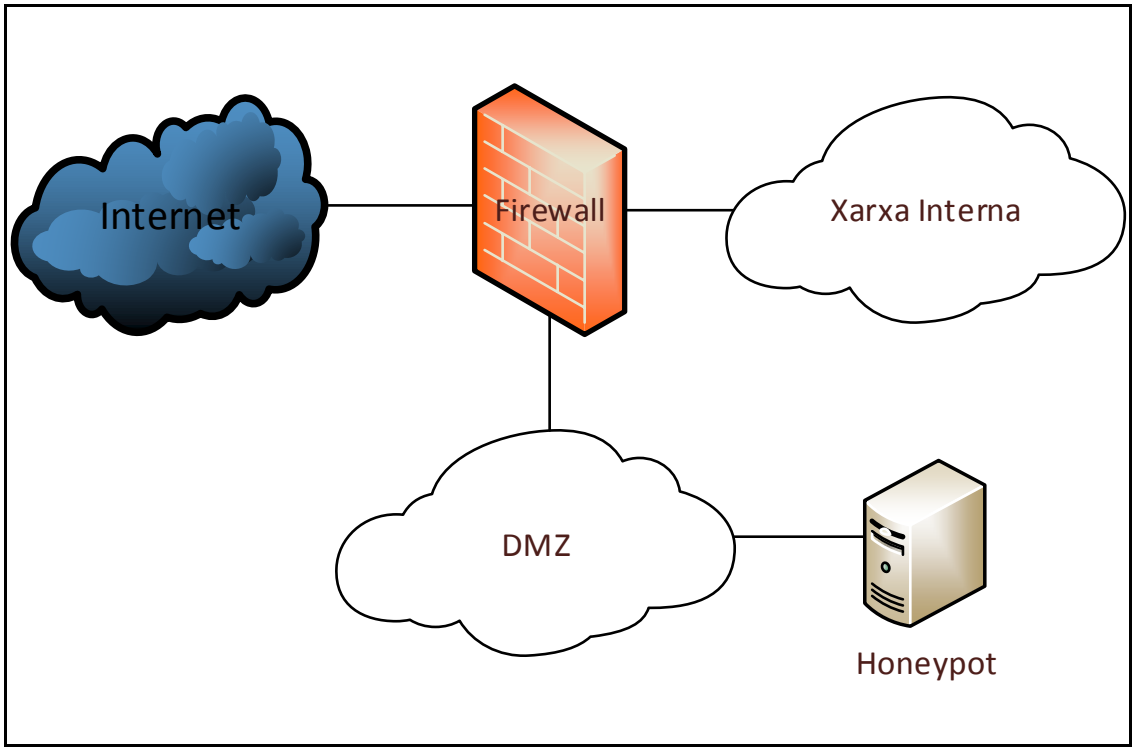

Fig 5: Topologia de xarxa amb DMZ

En el nostre cas com no disposem de cap infraestructura pre-fixada i com els requisits del projecte marquen que el nostre objectiu és captar el màxim número d'atacs. Lo més normal seria situar el nostre honeypot davant del Firewall, però també existeix la possibilitat de posar-ho darrera del Firewall fent NAT de tots els serveis que desitgem que siguin atacats i permitent el tràfic en aquests ports. Amb aquesta segona opcion podem preservar l'accés a les consoles de Honeypot des de l'exterior.

5.2.3. Selecció de tecnologies

Com recomana l'enunciat del projecte la implementació dels honeypots que farem estarà basada en la distribució Honeydrive. Aquesta distribució ens ofereix diferents solucions tecnològiques per a implementar un honeypots. Pel que fa a honeypots de mitja interacció trobem principalment tres opcions Honeyd, Amun i Dionaea per a implementar honeypots generalistes que permeten emular Linux i Windows, i els diferents serveis estàndard que funcionen sobre cadascuna de les plataformes (MySQL, MsSQL, Apache, IIS, Exchange, ...).

Les tres opcions ens donen funcionalitats molts similars a l'hora de fer honeypots, però Dionaea destaca junt amb Kippo (honeypot SSH) per tenir una interfície gràfica que facilita la anàlisis posterior dels atacs. Per tant de entre les opcions disponibles hem seleccionat:

- Dionaea ens dona gràfics i estadístiques dels orígens dels atacks per països, per IP, etc.

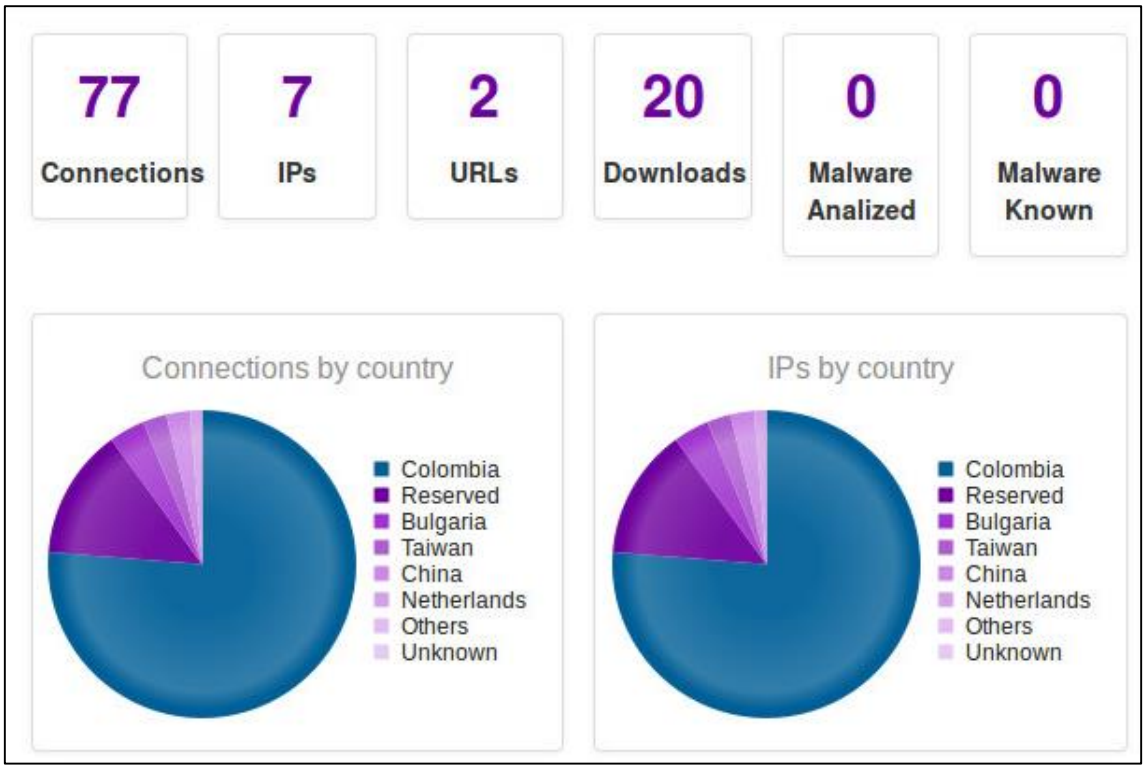

Fig 6: Dionaea, exemple resum per país

ID	Type	Transport	Protocol	Date	Root	Parent	Sensor	DST Port	Attacker	Hostname	SRC Port
171	accept	tcp	smbd	04-05-2014 20:13:07	171	—	192.168.0.100	445	190.69.199.166	—	1483
170	reject	tcp	pcap	04-05-2014 20:13:07	170	—	192.168.0.100	139	190.69.199.166	—	1484
169	accept	tcp	smbd	04-05-2014 20:13:06	169	—	192.168.0.100	445	190.69.199.166	—	1436
168	accept	tcp	smbd	04-05-2014 20:12:59	168	—	192.168.0.100	445	190.69.199.166	—	1033
167	reject	tcp	pcap	04-05-2014 20:12:59	167	—	192.168.0.100	139	190.69.199.166	—	1034
166	accept	tcp	smbd	04-05-2014 20:12:58	166	—	192.168.0.100	445	190.69.199.166	—	4896

Fig 7: Dionaea, exemple connexions rebudes

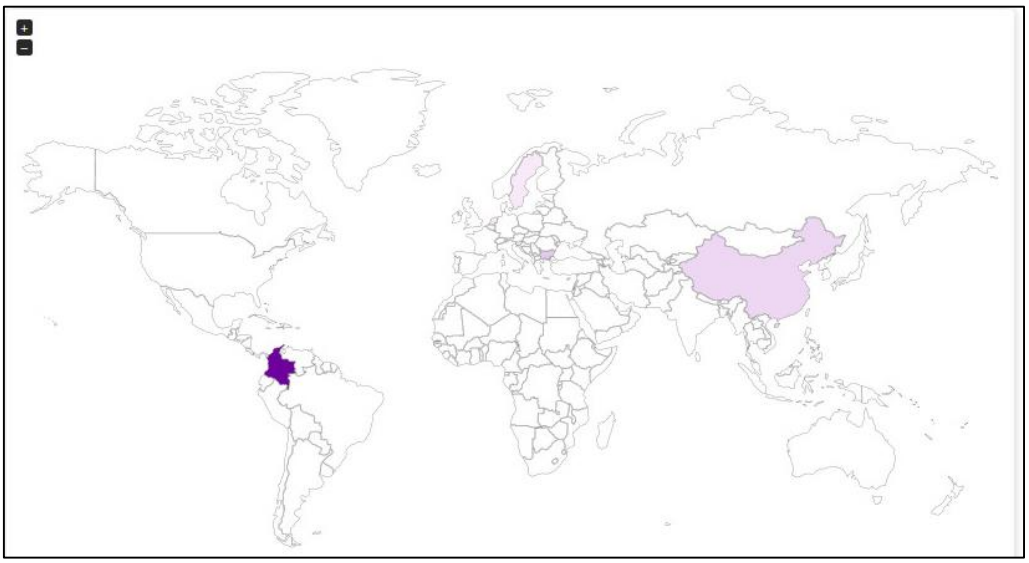

Fig 8: Dionaea, exemple geolocalització dels atacs

- Kippo (ssh emulator) ens dona diferents gràfics i estadístiques amb informació general sobre el número de intents de login, IPs d'origen, convinació user:pass més utilitzades, logins amb èxit, etc.

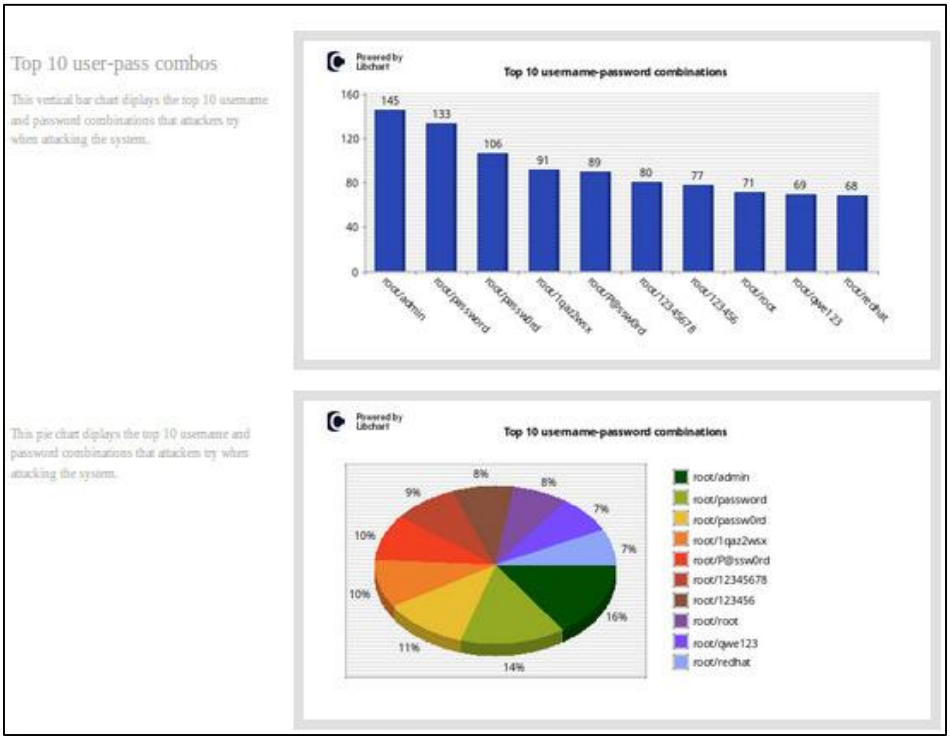

Fig 9: Kippo, exemple convinacions user:pass explotades

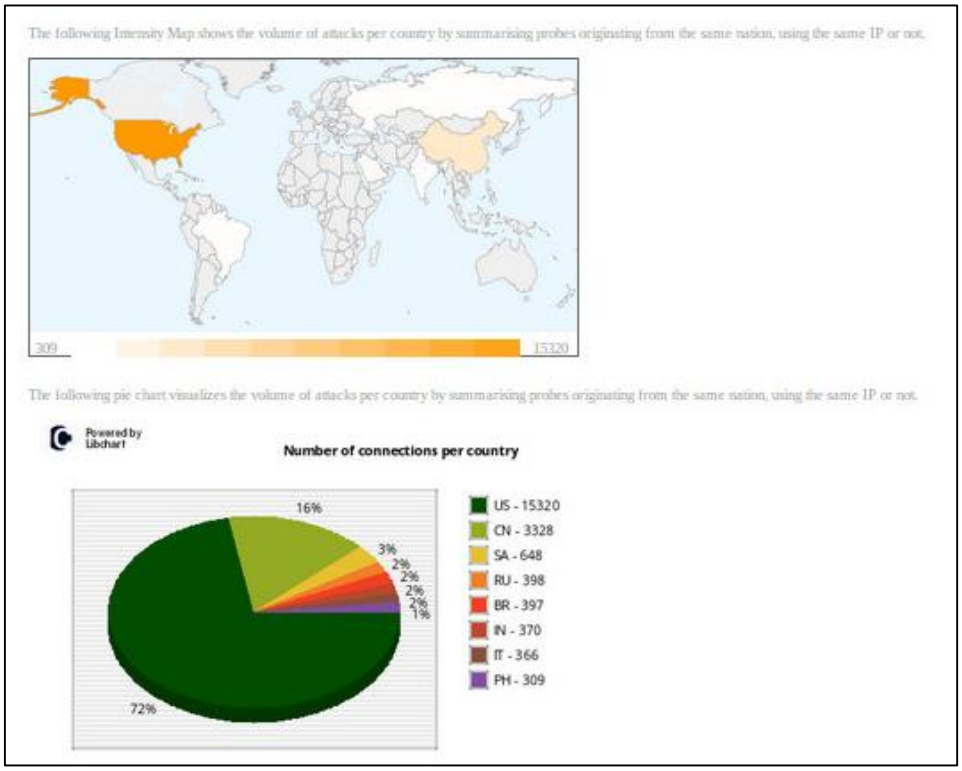

Fig 10: Kippo, exemple geolocalització de l'atac

S'han seleccionat aquestes solucions tecnològiques perquè són les que més s'adapten a les necessitats del projecte. Ja que, ens proporcionen uns honeypots de mitja interacció que ens permeten emular entorns tant Linux com Windows a l'hora que ens ofereix una sèrie d'eines que ens simplifica la recol·lecció i l'anàlisi dels atacs rebuts. Això junt amb la opció de emular entorns windows ens simplificarà molt l'anàlisi posterior fer-nos estalviar temps, esforç, etc i ens permet no haver d'implementar un entorn Windows ni el software necessari per fer el posterior anàlisi d'atacs rebuts sobre el sistema.

Pel que fa a l'entorn on es muntarà el honeypot, s'ha aprofitant que desenvolupo la meva activitat professional en una empresa de l'entorn tecnològic i aquesta m'ha brindat l'oportunitat de implementar la meva solució honeypot sobre el seus sistemes. Més concretament sobre un entorn cloud. Per tant, per implementar les xarxes i la màquina virtual que allotjarà el honeypot es farà servir vCloud de VMWare. Aquest entorn ens brinda més opcions que la opció suggerida al l'enunciat a l'hora que ens assegura certa estabilitat durant tot el procés.

A banda de tot lo relacionat directament amb la infraestructura dels honeypots que implementarem, també s'ha considerat important poder tenir mètriques del servidor. Perquè si les contrastem amb les dades recollides amb el honeypot ens poden donar informació extra sobre els recursos consumits per la màquina en un moment concret

6. Disseny del sistema

En aquest punt es tractaran amb detall les diferents decisions tècniques que s'han pres durant la implementació del honeypot, així com una visió global del projecte vist des de l'àrea tècnica.

6.1. Disseny detallat del sistema

6.1.1. Arquitectura de l'entorn virtual

Com s'ha comentat en capítols previs, l'entorn s'ha muntat sobre un entorn cloud públic, més concretament sobre vCloud 5.1 de VMWare. Aquesta decisió ens ha facilitat la implementació del projecte a l'hora que ens ha donat una gran flexibilitat per dimensionar la màquina i implementar la arquitectura de xarxa.

L'estructura bàsica d'un entorn cloud està formada per quatre elements: l'organització, el virtual data center (vDC), la virtual application (vAPP) i el màquina virtual (VM).

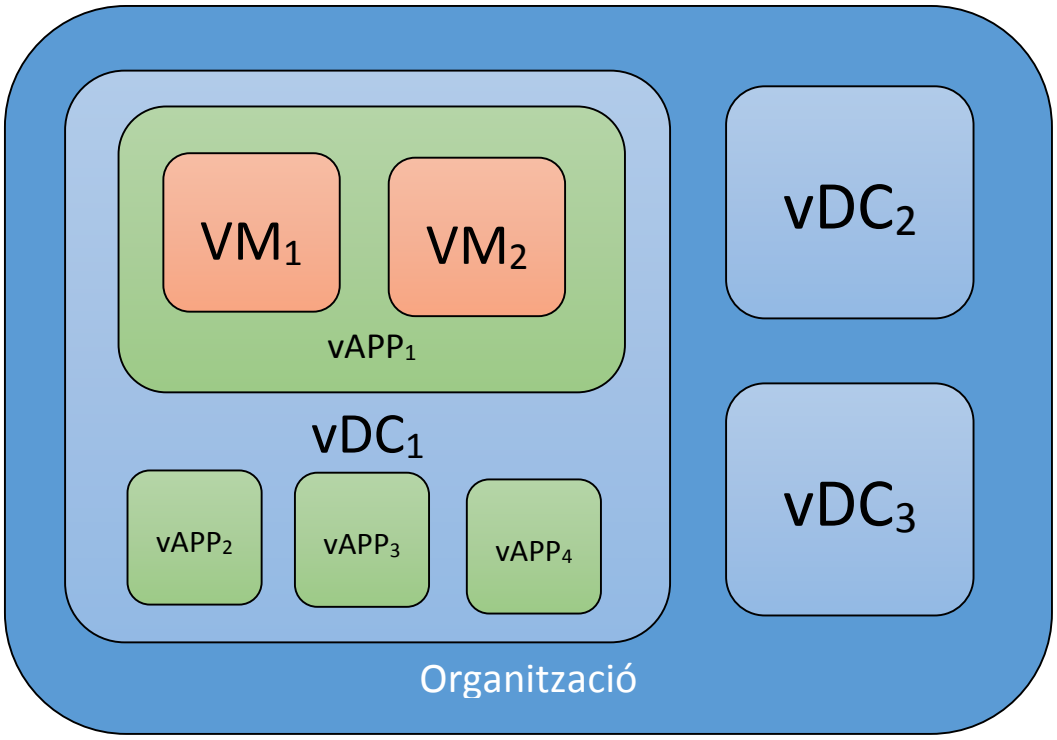

Fig 11: vCloud, exemple d'estructura virtual

La organització és el contenidor que separa a nivell lògic els nostres sistemes dels de la resta de client. Dins d'aquest trobem el que s'anomena virtual data center i podem tenir un o més.

Un virtual data center és el contenidor on es defineixen els recursos dels que disposarà el nostre entorn cloud i que pot contenir una o més vAPPs. Entenent per recursos la quantitat de CPU, RAM i HDD que tindran disponibles les virtual Applications que aculli.

Una vAPP és la unitat mínima que es pot definir sobre vCloud i pot estar formada per una o més màquines virtuals. Un exemple pot ser una vAPP d'una web que conté dues màquines una VM amb un frontal web i una altre VM amb una base de dades.

En el nostre cas, per implementar la solució em creat una Organització independent sobre la qual hem disposat d'un virtual datacenter amb recursos limitats que allotja una virtual application (vAPP) que conté la màquina virtual (VM) que allotja el honeypot.

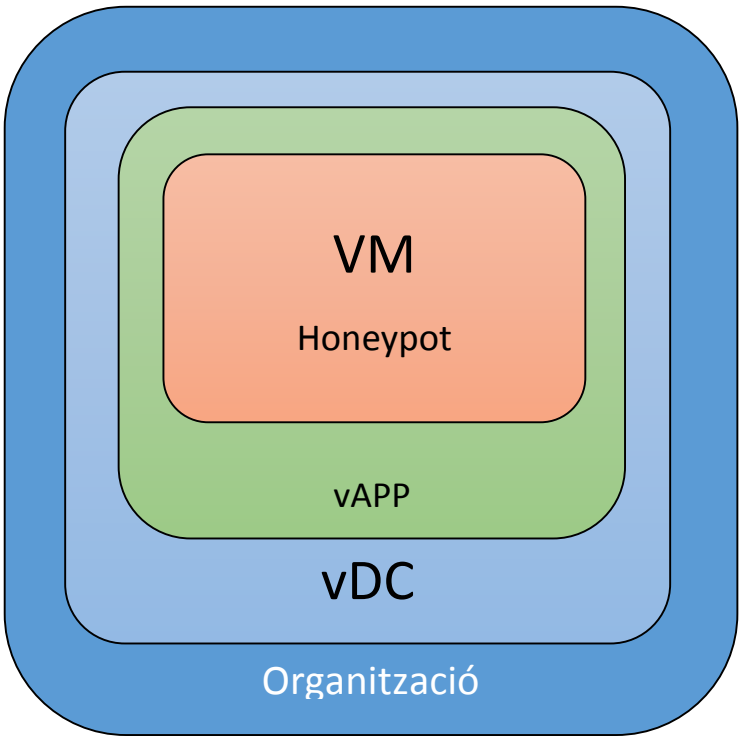

Fig 12: vCloud, estructura virtual del honeypot

6.1.2. Arquitectura de xarxes

A nivell de xarxes s'ha optat per una xarxa plana en la qual tenim un Firewall entre Internet i el nostre honeypot. S'ha decidit instal·lar un Firewall perquè el honeypot publica serveis que ens seran útils per l'anàlisi dels atacs, però que no han de ser visibles des de Internet. Més explícitament, fem servir el Firewall per amagar d'internet les interfícies webs que ens donen les estadístiques i la base de dades que allotja informació sobre els atacs. De manera que hem fet un NAT 1:1 on redirigim tots els ports de la IP pública W.X.Y.Z[1] a la IP privada del honeypot 192.168.0.100 on hem tancat els ports 8000 (port per on es publiquen les estadístiques de DionaeaFR), 8888 (port per on es publiquen les estadístiques de kippo graph) i 33006 (base de dades on es guarden les dades de kippo).

[1] S'ha decidit no descriure la IP pública del sistema perque pertany a un rang delegat a la entitat que ens ha facilitat l'accés a l'entorn cloud.

ID de regla	Nombre	Origen	Destino	Protocolo	Direcci...
1	Deny kippoGraph	Cualquiera:Cualquiera	Cualquiera:8888	TCP	Entrante
2	Deny DionaeaFR	Cualquiera:Cualquiera	Cualquiera:8000	TCP	Entrante
3	Allow all	Cualquiera:Cualquiera	Cualquiera:Cualquiera	CUALQUIEF	Entrante
4	Allow all outgoing	Cualquiera:Cualquiera	Cualquiera:Cualquiera	CUALQUIEF	Saliente
5	Deny MySQL 3300	Cualquiera:Cualquiera	Cualquiera:33006	TCP	Entrante

Fig 13: Regles de firewall

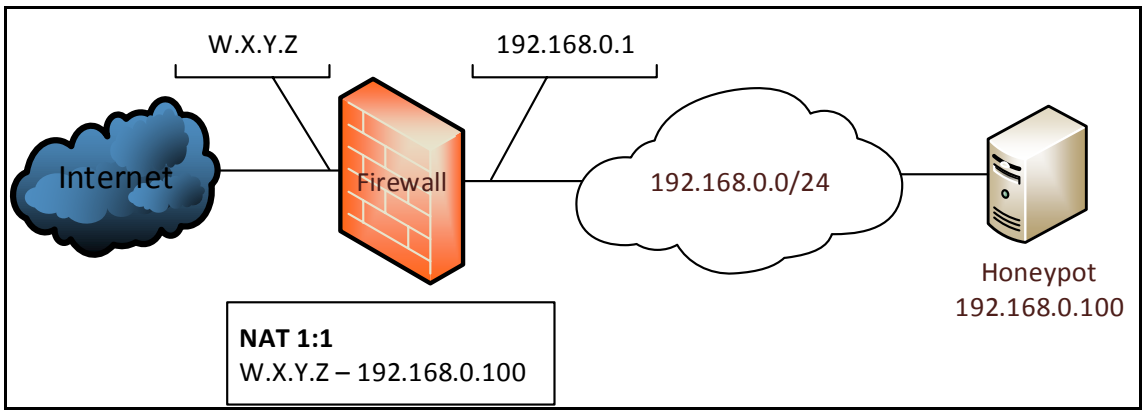

Fig 14: Topografia de xarxa del honeypot

Per implementar el Firewall s'ha fet servir la tecnologia de Firewall virtual que en ofereix VMWare al seu entorn cloud, vShield. El qual disposa de una IP pública que com s'ha descrit anteriorment està natejada a la IP privada del honeypot i una IP privada dins de la xarxa 192.168.0.0/24.

6.1.3. Arquitectura del honeypot

Com s'ha citat anteriorment, per implementar les honeypot s'han escollit dues tecnologies que faciliten la recol·lecció de dades a l'hora que permeten emular honeypot basats en Windows i Linux. Per evitar, d'aquesta manera, implementar tot un sistema basar en el S.O. de Microsoft.

Kippo

Com s'ha descrit en apartats anteriors Kippo és un honeypot que simula ser un servei SSH (port 22 TCP) complert que deixa interactuar mitjanament a l'atacant. Nosaltres l'hem implementat junt a un complement anomenat Kippo Graph que ens dóna gràfiques i estadístiques dels atacs.

Des de un principi la idea era que la màquina que hem muntat faci a l'hora de honeypot per Windows i Linux a l'hora. Això va provocar conflictes entre els serveis necessaris per a implementar i fer servir Kippo. Els problemes venien donats perquè Dionaea simulava honeypot de Apache i MySQL als seus ports estàndard, això entrava en conflicte amb els serveis que fa servir Kippo per guardar els seus registres (MySQL) i el servei que dona accés a Kippo Graph. Per solucionar-ho, s'ha fet que aquestes dues eines facin escoltar els seus serveis per altres ports. En el nostre cas s'ha fet servir el port 3306 per al MySQL i 8888 pel servei Apache. Per aquest motiu aquests ports estan filtrats al Firewall que separa el honeypot d'Internet.

Pel que fa a la configuració de Kippo s'ha optat per deixar-la per defecte a excepció de les convinacions de user:pass que donen accés al honeypot SSH (userdb.txt). A més, cal destacar que és molt importat no aixecar aquest honeypot amb l'usuari root. D'aquesta manera evitem comprometre la màquina sencera en cas de fallida al honeypot. Per aquest motiu al script d'arrancada de kippo, aquest s'aixeca amb l'usuari honeydrive[2].

Dionaea

En la selecció de tecnologies s'ha optat per Dionaea per la seva versatilitat per emular protocols Windows i Linux, i per contar amb el complement DionaeaFR que ens dóna un anàlisis detallat dels atacs rebuts.

En el nostre cas no s'ha variat la configuració per defecte, degut que en aquesta ja s'emulaven tecnologies Microsoft i Linux. Més concretament les tecnologies emulades han sigut les següents:

- Windows:
  - Micrososft SQL Server (MsSQL) (port 1433 TCP)
  - WINS (port 42 TCP)
  - NetBios (port 139 TCP)
  - Microsoft Active Directory / SMB (port 445 TCP)
- Linux:
  - HTTP (port 80 i 443 TCP)
  - FTP (port 21 TCP)
  - TFTP (port 69 UDP)
  - Telnet (Port 23 TCP)
  - MySQL (port 3306 TCP)
- Altres:
  - Sip (port 5060 , 5061 TCP i 5060 UDP)

Amb DionaeaFR ens vam trobar amb un petit problema, aquest servei de reporting aixeca un petit servidor web al port 8000 per on mostra les estadístiques dels atacs rebuts. Per això, com en el cas de Kippo Graph, es va decidir tancar l'accés a aquest port.

La configuració detallada dels dos sistemes de honeypot es troben completament a l'annex 1, junt amb els scripts creats per arrencar i aturar els honeypot i/o els complements per generar les estadístiques.

7. Recol·lecció de dades

Per tal d'aconseguir una base de dades sobre la que realitzar un bon informe sobre els atacs rebuts, s'ha deixat el honeypot escoltant connexions amb tots els serveis esmentats al punt 6.1.3 durant 15 dies de forma ininterrompuda, del 5 al 23 de maig.

Totes les dades recol·lectades durant aquest període es poden veure a l'annex 2. Sobre aquestes dades es recolza l'informe del punt 8 i les conclusions del projecte.

[2] Veure annex 1 punt 10.1.2 Script arrancada kippo kippo_start.sh

8. Informe dels atacs

El passat dilluns 5 de maig de 2014 es van i exposar a Internet els honeypot. Aquests van estar encesos durant 15 dies en els quals van rebre contínuament atacs.

En el transcurs d'aquest temps s'han tinguts exposats serveis basats tant en Linux com en Microsoft Windows[3] i com podem veure a la fig 15 s'han rebut connexions contínuament.

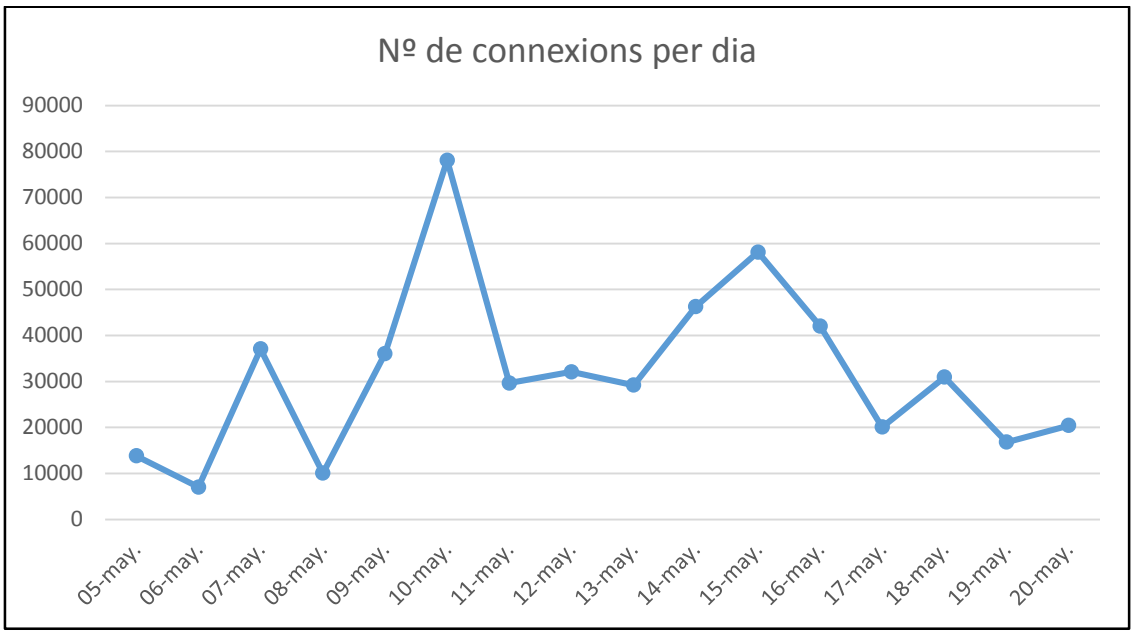

Fig 15: Nº de connexions per dia

L'origen d'aquestes connexions han estats distribuïts per tot el món, però les països que més activitat han tingut són Rússia, els Estats Units i Alemanya. També cal destacar que hem tingut atacs amb origen desconeguts o reservat.

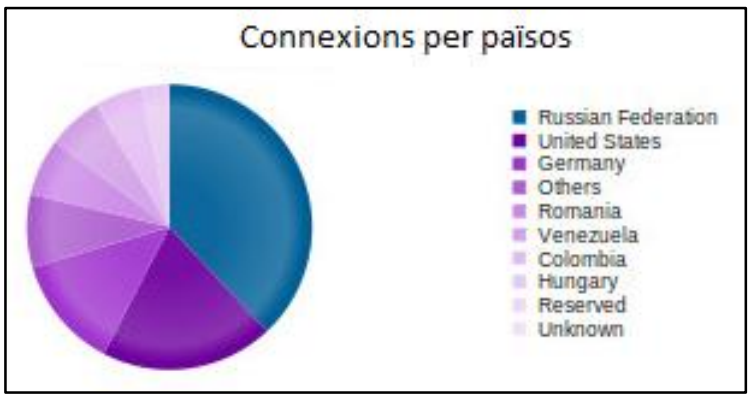

Fig 16: Connexions per països

[3] Es poden veure detalalts al subapartat Dionaea del punt 6.1.3

L'objectiu les connexions ha sigut principalment protocols o servies fets servir normalment per sistemes operatius Microsoft Windows, rebent el 84% de les connexions (resultat de sumar SMB i NetBios). Mentre que els protocols fets servir per Linux (principalment SSH) han sigut objectiu del 12% de les connexions.

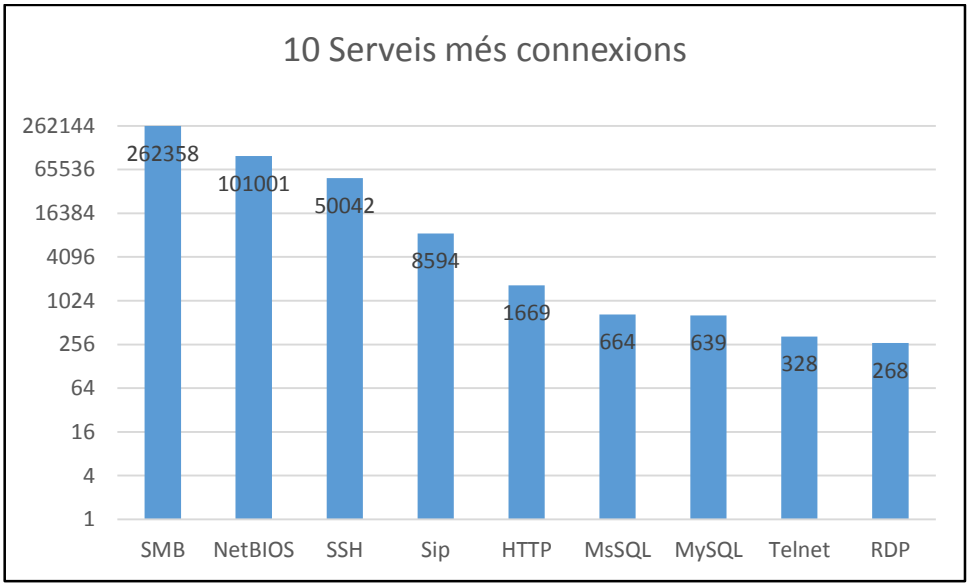

Fig 17: gràfica del 10 serveis més connexions

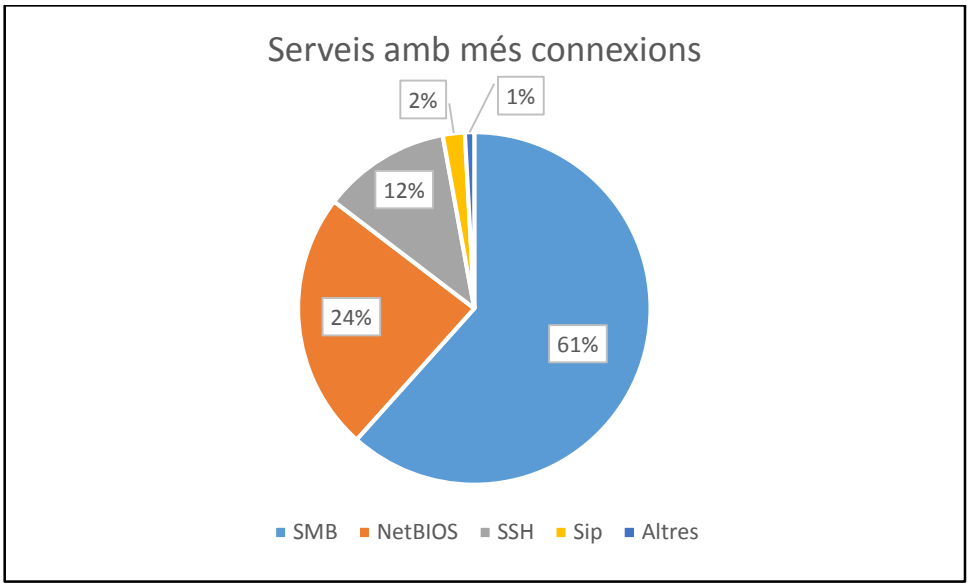

Fig 18: gràfica de distribució dels 10 serveis amb més connexions.

Pel que fa als atacs, aquest s'han distribuït entre 5 serveis principalment SMB, SSH, HTTP, MsSQL i MySQL, seguint la distribució entre serveis de la figura 19. A primera vista es pot veure com la gran majoria dels atacs van contra SMB i SSH mentre que la resta de serveis no presenten atacs o no són representatius.

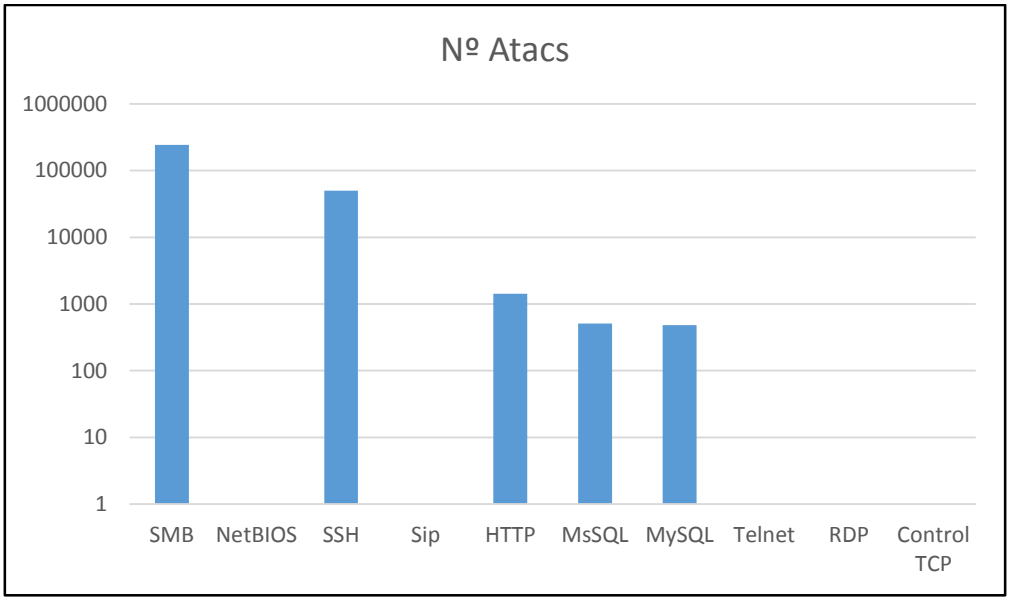

Fig 19: Nº d'atacs per servei

Si comparem els atacs, com era d'esperar els serveis amb més connexions són els més atacats. Però trobem excepcions en serveis amb moltes connexions, com per exemple NetBIOS que després no ha rebut cap atac.

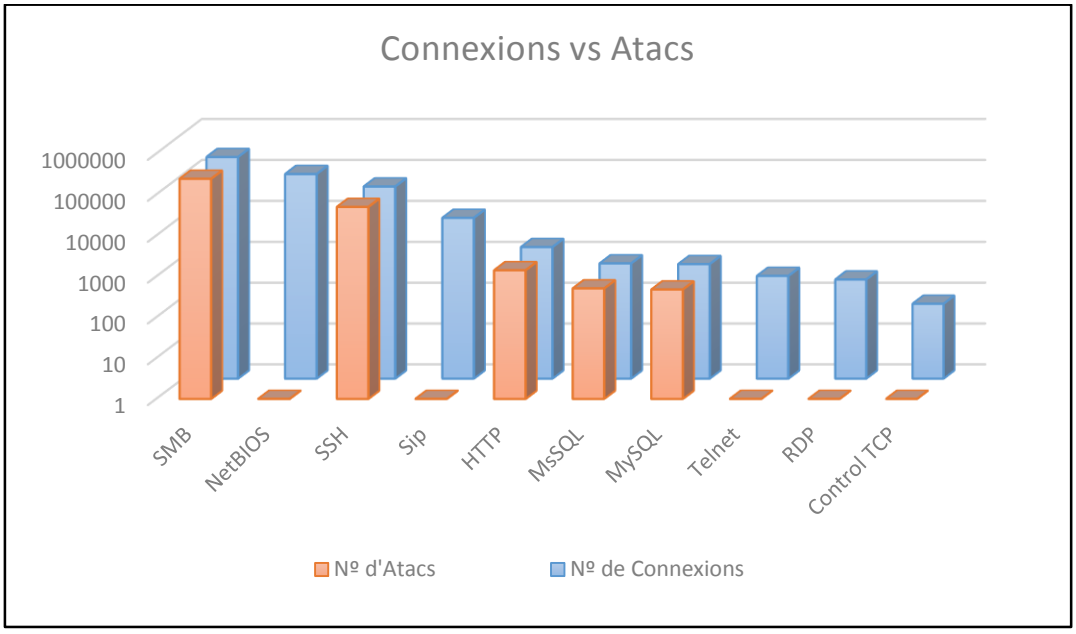

Fig 20: Connexions vs Atacs

En el gràfic de la figura 21, si fem la correlació de servei SMB igual a Windows i servei SSH igual a Linux, podem apreciar ràpidament com els sistemes Windows són molt més atacats que els sistemes basats en Linux amb un ratio 1:5, per cada atac a Linux Windows en rep 5.

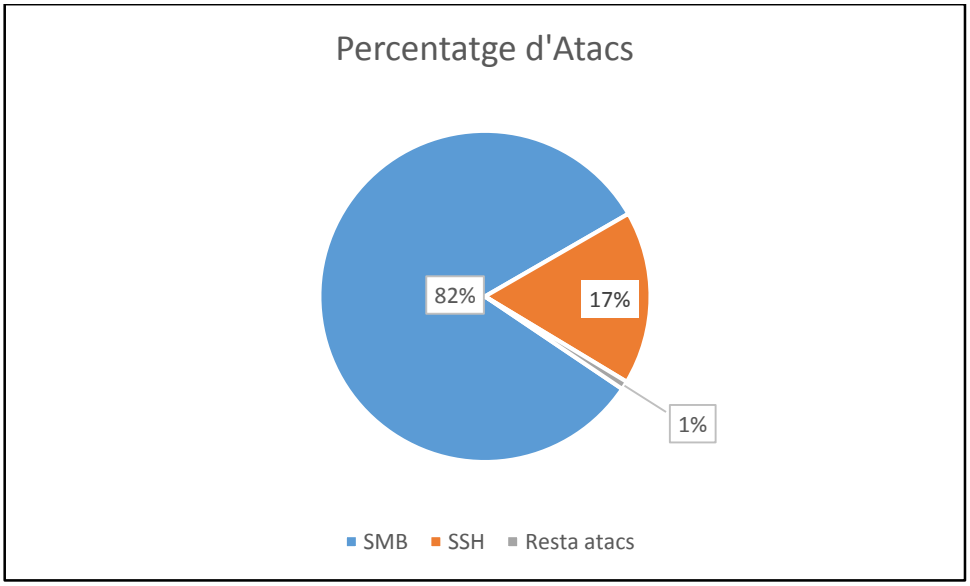

Fig 21: Persentatge d'atacs

Per finalitzar, al analitzar els serveis amb més percentatge d'atac veiem com el mètode més empleat per atacar SSH és intentar esbrinar usuari i pasword fent servir força bruta basada en diccionari. Mentre que pel que fa al servei SMB es sol fer servir la vulnerabilitat MS08-067[4].

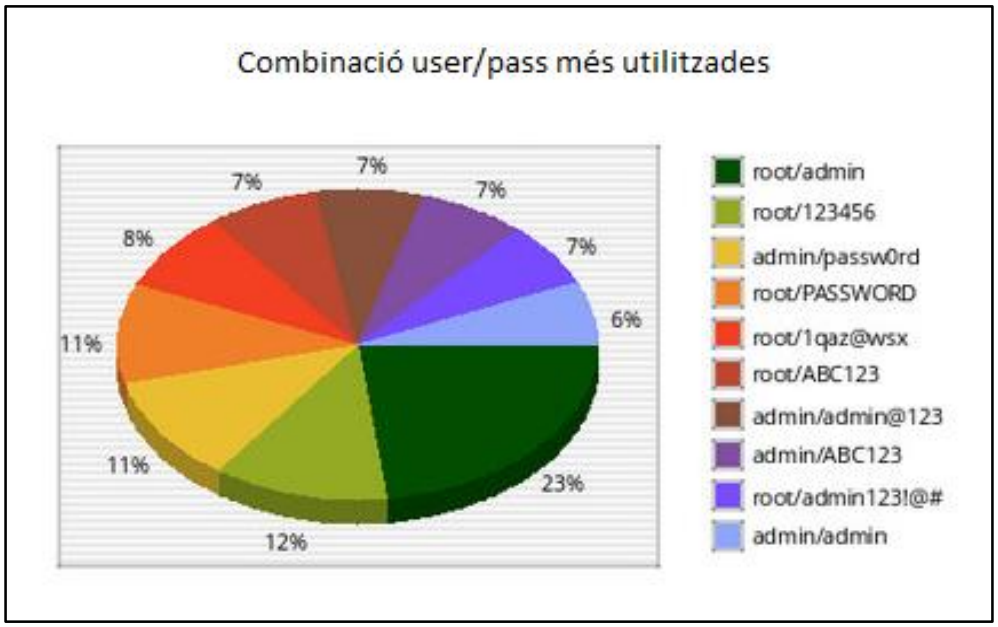

Fig 22: Combinació user/pass més utilitzades per atacar SSH

[4] https://technet.microsoft.com/es-es/library/security/ms08-067.aspx

9. Conclusions

Un cop finalitzat l'informe en el que s'analitza tant les connexions com els atacs rebuts arribem a la conclusió, de que hi ha milers d'aranyes a Internet que no només es dediquen a indexar el contingut web, com pot fer Google o Bing, sinó que es dediquen a realitzar anàlisis de ports per localitzar serveis amb deficiències de seguretat. Aquestes deficiències seran fetes servir per l'atacant amb l'objectiu de comprometre les màquines vulnerables.

Una altra conclusió a la que arribem es que els atacs fets per les aranyes descrites al paràgraf anterior tenen com objectiu atacar indiferentment entorns Linux i Windows. Però en el nostre cas concret, els serveis emulats basats en Linux han patit molts menys atacs que els basats Windows, amb una proporció d'un atac contra Linux per cada 5 atacs a Windows.

10. Annexes

10.1. Annex 1: Configuració del honeypot

10.1.1. Script arrancada Dionaea dionaea_start.sh

```
#!/bin/bash

fecha=$( date '+%y%m%d%H%M' )

echo Starting dionaea

cd /opt/dionaea/bin
./dionaea -l all,-debug -L '*' > /var/log/dionaea/dionaea_$fecha.log 2>
/var/log/dionaea/dionaea_error_$fecha.log &

exit 0
```

10.1.2. Script arrancada kippo kippo_start.sh

```
#!/bin/bash

fecha=$( date '+%y%m%d%H%M' )

su honeydrive -c "cd /opt/kippo && ./start.sh > /var/log/kippo/kippo_$fecha.log
2> /var/log/kippo/kippo_error_$fecha.log" &

exit 0
```

10.1.3. Script arrancada DionaeaFR dionaeaFR_start.sh

```
#!/bin/bash

echo Starting dionaeaFR
cd /opt/dionaeaFR
python manage.py collectstatic
python manage.py runserver 192.168.0.100:8000
```

10.1.4. Script aturada Dionaea dioanea_stop.sh

```
#!/bin/bash

echo stoping dionaea

for pid in $( ps -ef | grep dionaea | egrep -v grep\|dionaea_stop.sh | cut -d '
' -f 7 ); do
       echo Matando $pid
       ps -ef | grep $pid |head -1
       kill -term $pid
done

echo dionaea stoped
echo process dionaea:
ps -ef | grep dionaea | egrep -v grep\|dionaea_stop.sh
echo -e "\n"
```

10.1.5. Script aturada Kippo kippo_stop.sh

```
#!/bin/bash

if [ -f /opt/kippo/kippo.pid ]; then
	pid=`cat /opt/kippo/kippo.pid`
	kill -term $pid
	echo kippo parado
	ps -ef | grep $pid | grep -v grep
	exit 0
else
	echo kippo no estaba arrancado
	exit 1
fi
```

10.1.6. Dionaea.conf

```
logging = {
	default = {
		// file not starting with / is taken relative to LOCALESTATEDIR
(e.g. /opt/dionaea/var)
		file = "log/dionaea.log"
		levels = "all,-debug"
		domains = "*"
    }

	errors = {
		// file not starting with / is taken relative to LOCALESTATEDIR
(e.g. /opt/dionaea/var)
		file = "log/dionaea-errors.log"
		levels = "warning,error"
		domains = "*"
	}
}

processors =
{
	filter-emu =
	{
		config = {
			allow = [{ protocol = 
["smbd","epmapper","nfqmirrord","mssqld"] }]
		}
		next = {
			emu =
			{
				config = {
					emulation = {
						limits = {
							files = "3"
							filesize = "524288" // 512 * 1024
							sockets = "3"
							sustain = "120"
							idle = "30"
							listen = "30"
							cpu = "120"
							steps = "1073741824" // 1024 *
1024 * 1024
						}

						/**
```

```
                                        * api default arguments for development
                                        * disabled by default
                                        * not working yet
                                        */
                                       api = {
                                              connect = {
                                                     host = "127.0.0.1"
                                                     port = "4444"
                                              }
                                       }
                                }
                        }
                }
        }

        filter-streamdumper =
        {
                config = {
                        allow = [
                                { type = ["accept"] }
                                { type = ["connect"] protocol=["ftpctrl"] }
                        ]
                        deny = [
                                { protocol = ["ftpdata", "ftpdatacon","xmppclient"] }
                        ]
                }
                next = {
                        streamdumper = {
                                config = {
                                        path = "var/dionaea/bistreams/%Y-%m-%d/"
                                }
                        }
                }
        }

/*      filter-sessions =
        {
                config = {
                        allow = [ { protocol = ["ftpctrl","remoteshell"] } ]
                }
                next = {
                        python = {
                                incident = "true"
                        }
                }
        }
*/
}

downloads =
{
        dir = "var/dionaea/binaries"
        tmp-suffix = ".tmp"
}

bistreams =
{
        python =
        {
                dir = "var/dionaea/bistreams"
        }
}
```

```
submit =
{
       defaults = {
              urls = ["http://anubis.iseclab.org/nepenthes_action.php",

       "http://onlineanalyzer.norman.com/nepenthes_upload.php",
                           "http://luigi.informatik.uni-
mannheim.de/submit.php?action=verify"]
              email = "nepenthesdev@gmail.com"
              file_fieldname = "upfile"
              MAX_FILE_SIZE = "1500000"
              submit        = "Submit for analysis"
       }

       /**
        * joebox is special, due to the TOS you can lookup here
        * http://www.joebox.org/resources/service%20terms.txt
        * therefore untested and disabled by default
        */
/*
       joebox = {
              urls = ["http://analysis.joebox.org/submit"]
              email = "nepenthesdev@gmail.com"
              file_fieldname = "upfile"
              MAX_FILE_SIZE = "1500000"
              submit        = "Submit for analysis"
              service       = "agree"
              xp            = "1"
              vista         = "1"
              w7            = "1"
              pcap          = "1"
       }
*/

/*
       yoursection =
       {
              urls = ["http://127.0.0.1/submit"]
              email = "yourmail"
              user = "yourusername"
              pass = "yourpassword"
       }
*/
}

listen =
{
       /* basically we have 3 modes
          - getifaddrs - auto
            will get a list of all ips and bind a service to each ip
          - manual - your decision
            addrs has to be provided, and should look like this
               addrs = { eth0 = ["1.1.1.1", "1.1.1.2"], eth1 = ["2.1.1.1",
"2.1.1.2"] }
              you get the idea ...
              for most cases with more than one address
              addrs = { eth0 = ["0.0.0.0"] }
              will do the trick
              if you want to throw in ipv6 support as well ...
              addrs = { eth0 = ["::"] }
              note: ipv6 does not work with surfids yet,
```

```
              as ipv6 addresses are mapped to ipv4 and surfids fails to retrieve
the sensor id for ::ffff:1.2.3.4
          - nl, will require a list of interfaces
            fnmatch is possible like
              interfaces = ["ppp*","tun*"]
              and loading the nl module AFTER the python module in the modules
section below
              nl will use the kernel netlink interface to figure out which
addresses exist
              at runtime, and start/stop services dynamically per address per
interface
       */

       mode = "getifaddrs"
       //addrs = { eth0 = ["0.0.0.0"] }
}

modules = {

       curl =
       {
              protocol = "http"
       }

       emu = {
              detect = "1"
              profile = "1"
       }

       pcap =
       {
              /**
               * libpcap 1.0.0
               *
               * "Arithmetic expression against transport layer headers, like
               * tcp[0], does not work  against  IPv6  packets. It  only looks
               * at IPv4 packets."
               *
               * As a consequence, the default filter can not match
               * ipv6 tcp rst packets.
               *
               * If you want to go for rejected ipv6, remove the tcp matching
part of the filter
               * The code is capable to checking the tcp-rst flag and seq number
itself, but
               * matching every packet in userspace is expensive.
               * Therefore you'll have to hack the code if you want to track ipv6
rejected connections
               *
               * Format is IFACE = { addrs = MODE }
               * currently mode is ignored
               */

              any = {
                     addrs = "auto"
              }
       }

       nfq =
       {
              /**
               * queue has to be the nfqueue num
               * refer to http://dionaea.carnivore.it/#nfq_python
```

```
                * if you do not specify a queue-num with iptables, 0 is the default
                */
               queue = "0"
       }

       python = {
               // default expands to PREFIX/lib/dionaea/python/
               // ordering is granted
               // useful for development
               // simply add your devel directory to the list, avoids a make
install for new python code
               sys_path = ["default"]

               // python imports
               imports      = [    "log",
                                   "services",
                                   "ihandlers"]
               ftp = {
                       root = "var/dionaea/wwwroot"

                       /* ftp client section
                        */

                       /* ports for active ftp
                        * string indicating a range
                        */
                       active-ports = "63001-64000"

                       /* host for active ftp via NAT
                        * 0.0.0.0 - the initiating connection ip is used for active
ftp
                        * not 0.0.0.0 - gets resolved as hostname and used
                        */
                       active-host = "0.0.0.0"
               }
               tftp = {
                       root = "var/dionaea/wwwroot"
               }
               http = {
                       root = "var/dionaea/wwwroot"
                       max-request-size = "32768" // maximum size in kbytes of the
request (32MB)
               }
               sip = {
                       udp = {
                               port = "5060"
                       }
                       tcp = {
                               port = "5060"
                       }
                       tls = {
                               port = "5061"
                       }
                       users = "var/dionaea/sipaccounts.sqlite"
                       rtp = {
                               enable = "yes"
                               /* how to dump the rtp stream
                                  bistream = dump as bistream
                               */
                               mode = ["bistream", "pcap"]

                               pcap = {
```

```
                                path  =  "var/dionaea/rtp/{personality}/%Y-%m-
%d/"
                                filename                                        =
"%H:%M:%S_{remote_host}_{remote_port}_in.pcap"
                        }
                }
                personalities = {
                        default = {
                                domain = "localhost"
                                name = "softphone"
                                personality = "generic"
                        }
                        /*
                        next-server = {
                                domain = "my-domain"
                                name = "my server"
                                personality = "generic"
                                serve = ["10.0.0.1"]
                                default_sdp = "default"
                                handle   =   ["REGISTER",   "INVITE",   "BYE",
"CANCEL", "ACK"]
                        }

                        */
                }
                actions = {
                        bank-redirect = {
                                do = "redirect"
                                params = {
                                }
                        }
                        play-hello = {
                                do = "play"
                                params = {
                                        file = "var/dionaea/.../file.ext"
                                }
                        }
                }
        }
        surfids = {
                sslmode = "require"
                host = "surfids.example.com" // change this
                port = "5432"                // maybe this
                username = "surfids"         // this
                password = "secret"          // and this
                dbname = "idsserver"
        }
        virustotal = {
                apikey = "........." // grab it from your virustotal account
at My account -> Inbox -> Public API
                file = "var/dionaea/vtcache.sqlite"
        }
        mwserv = {              // ask your mwserv backend provider for
needed values
                url = ""        //  the  url  to  send  the  submission
requests to
                maintainer = "" // username of the maintainer of this
sensor
                guid = ""       // guid of this sensor, as generated
serverside; typically 8 chars
                secret = ""     // shared secret used for authentication
aka password; typically 48 chars
        }
```

```
		mysql = {
			databases = {
				information_schema = {
					path = ":memory:"
				}

				// example how to extend this
				//  just  provide  a  databasename  and  path  to  the
database
				// the database can be altered by attackers, so ...
better use a copy
//				psn = {
//					path = "/path/to/cc_info.sqlite"
//				}

			}
		}
		submit_http = {                  // ask your submit_http backend
provider for needed values
			url = ""            //  the  url  to  send  the  submission
requests to
			email = ""          // optional
			user = ""           // username (optional)
			pass = ""           // password (optional)
		}
		logsql = {
			mode = "sqlite" // so far there is only sqlite
			sqlite = {
				file = "var/dionaea/logsql.sqlite"
			}
		}
		logxmpp = {
			/**
			 * this section defines a single xmpp logging target
			 * you can have multiple
			 */
			carnivore = {
				server = "sensors.carnivore.it"

				/**
				 * as dionaea does not support starttls (xmpp on port
5223),
				 * we rely on 'legacy ssl' for the xmpp connection
(port 5222)
				 */
				port = "5223"
				muc = "dionaea.sensors.carnivore.it"

				/**
				 * if the server exists, this is a valid account
				 */
				username = "anonymous@sensors.carnivore.it"
				password = "anonymous"

				/**
				 * setting a resource is possible, but you should not
do it
				 * the default resource is a random string of 8 chars
				 */
				// resource = "theresource"
				config =
				{
					/**
```

```
                                * this defines a muc channel
                                */
                               anon-events =
                               {
                                      /**
                                       * incidents matching these events will
get relayed to the channel
                                       */
                                      events                                   =
["^dionaea\x5c.connection\x5c..*",

      "^dionaea\x5c.modules\x5c.python\x5c.smb.dcerpc\x5c.*",

      "^dionaea\x5c.download\x5c.offer$",

      "^dionaea\x5c.download\x5c.complete\x5c.hash$",

      "^dionaea\x5c.module\x5c.emu\x5c.profile$",

      "^dionaea\x5c.modules\x5c.python\x5c.mysql\x5c.*",

      "^dionaea\x5c.modules\x5c.python\x5c.sip\x5c.*"
                                                              ]

                                      /**
                                       *  anonymous  removes  the  local  host
information from all connection messages
                                       *  so  you  can  report  without  getting
identified
                                       */
                                      anonymous = "yes"
                               }

                               anon-files =
                               {
                                      events                                   =
["^dionaea\x5c.download\x5c.complete\x5c.unique"]
                               }
                        }
                 }
          }
          nfq = {
                 /**
                  * nfq can intercept incoming tcp connections during the tcp
handshake
                  * giving your honeypot the possibility to provide service
on
                  * ports which are not served by default.
                  *        refer        to        the        documentation
(http://dionaea.carnivore.it/#nfq_python)
                  * BEFORE using this
                  */

                 nfaction = "0" // DROP

                 throttle = {
                        window = "30"
                        limits = {
                               total = "30"
                               slot = "30"
                        }
                 }
```

```
                timeouts = {
                        server = {
                                listen = "5"
                        }
                        client = {
                                idle = "10"
                                sustain = "240"
                        }
                }
        }
        p0f = {
                /**
                 * start p0f with
                 * sudo p0f -i any -u root -Q /tmp/p0f.sock -q -l
                 */
                path = "un:///tmp/p0f.sock"
        }

        fail2ban = {
                downloads = "var/dionaea/downloads.f2b"
                offers = "var/dionaea/offers.f2b"
        }

        ihandlers = {
                handlers = ["ftpdownload", "tftpdownload", "emuprofile",
"cmdshell", "store", "uniquedownload",
                "logsql",
//              "virustotal",
//              "mwserv",
//              "submit_http",
//              "logxmpp",
//              "nfq",
                "p0f",
//              "surfids",
//              "fail2ban"
                ]
        }

        services = {
                serve = ["http", "https", "tftp", "ftp", "mirror", "smb",
"epmap", "sip","mssql", "mysql"]
        }

}

nl =
{
        lookup_ethernet_addr = "no" // set to yes in case you are interested
in the mac address  of the remote (only works for lan)

}


/* nc is a test module */
/*      nc =
{
        services = [
                {
                        proto = "redir"
                        type = "tcp"
                        host = "::"
                        port = "4711"
                },
```

```
{
	proto = "redir"
	type = "tcp"
	host = "::"
	port = "12344"
},
{
	proto = "sink"
	type = "tcp"
	host = "::"
	port = "12345"
	throttle = {
		in = "8192"
	}
	timeout = {
		listen = "15"
		connect = "15"
	}
},
{
	proto = "source"
	type = "tcp"
	host = "::"
	port = "12346"
	throttle = {
		out = "8192"
	}
	timeout = {
		listen = "15"
		connect = "15"
	}
},
{
	proto = "redir"
	type = "tcp"
	host = "::"
	port = "12347"
	throttle = {
		in = "8192"
		out = "8192"
	}
	timeout = {
		listen = "15"
		connect = "15"
	}
},
{
	proto = "redir"
	type = "tls"
	host = "::"
	port = "12444"
	timeout = {
		listen = "15"
		connect = "15"
	}
},

{
	proto = "sink"
	type = "tls"
	host = "::"
	port = "12445"
	throttle = {
```

```
                        in = "8192"
                }
                timeout = {
                        listen = "15"
                        connect = "5"
                }
        },
        {
                proto = "source"
                type = "tls"
                host = "::"
                port = "12446"
                throttle = {
                        out = "8192"
                }
                timeout = {
                        listen = "15"
                        connect = "15"
                }
        },
        {
                proto = "redir"
                type = "tls"
                host = "::"
                port = "12447"
                throttle = {
                        in = "8192"
                        out = "8192"
                }
                timeout = {
                        listen = "15"
                        connect = "15"
                }
        },
        {
                proto = "source"
                type = "udp"
                host = "::"
                port = "12544"
                timeout = {
                        connect = "15"
                }
        },
        {
                proto = "sink"
                type = "udp"
                host = "::"
                port = "12545"
                timeout = {
                        connect = "15"
                }
        },
        {
                proto = "redir"
                type = "udp"
                host = "::"
                port = "12546"
                timeout = {
                        connect = "15"
                }
        }
]
```

```
            clients = [
                  {
                        proto = "source"
                        type = "tcp"
                        host = "127.0.0.1"
                        port = "13344"
                        timeout = {
                              connecting = "5"
                              connect = "15"
                              reconnect = "5"
                        }
                  },
                  {
                        proto = "redir"
                        type = "tcp"
                        host = "ip6-localhost"
                        port = "13345"
                        timeout = {
                              connecting = "5"
                              connect = "15"
                              reconnect = "5"
                        }
                  },
                  {
                        proto = "redir"
                        type = "tls"
                        host = "localhost"
                        port = "13346"
                        timeout = {
                              connecting = "5"
                              connect = "15"
                              reconnect = "5"
                        }
                  },
                  {
                        proto = "source"
                        type = "tls"
                        host = "ip6-localhost"
                        port = "12445"
                        timeout = {
                              reconnect = "1"
                              connect = "1"
                        }
                  }
            ]
      }
 */
}
```

10.1.7. Kippo.cfg

```
#
# Kippo configuration file (kippo.cfg)
#

[honeypot]

# IP addresses to listen for incoming SSH connections.
#
# (default: 0.0.0.0) = any address
#ssh_addr = 0.0.0.0

# Port to listen for incoming SSH connections.
```

```
#
# (default: 2222)
ssh_port = 22

# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment.
#
# (default: sales)
hostname = webserver

# Directory where to save log files in.
#
# (default: log)
log_path = log

# Directory where to save downloaded (malware) files in.
#
# (default: dl)
download_path = dl

# Directory where virtual file contents are kept in.
#
# This is only used by commands like 'cat' to display the contents of files.
# Adding files here is not enough for them to appear in the honeypot - the
# actual virtual filesystem is kept in filesystem_file (see below)
#
# (default: honeyfs)
contents_path = honeyfs

# File in the python pickle format containing the virtual filesystem.
#
# This includes the filenames, paths, permissions for the whole filesystem,
# but not the file contents. This is created by the createfs.py utility from
# a real template linux installation.
#
# (default: fs.pickle)
filesystem_file = fs.pickle

# Directory for miscellaneous data files, such as the password database.
#
# (default: data_path)
data_path = data

# Directory for creating simple commands that only output text.
#
# The command must be placed under this directory with the proper path, such
# as:
#   txtcmds/usr/bin/vi
# The contents of the file will be the output of the command when run inside
# the honeypot.
#
# In addition to this, the file must exist in the virtual
# filesystem {filesystem_file}
#
# (default: txtcmds)
txtcmds_path = txtcmds

# Public and private SSH key files. If these don't exist, they are created
# automatically.
#
# (defaults: public.key and private.key)
public_key = public.key
private_key = private.key
```

```
# Initial root password. NO LONGER USED!
# Instead, see {data_path}/userdb.txt
#password = 123456

# IP address to bind to when opening outgoing connections. Used exclusively by
# the wget command.
#
# (default: not specified)
#out_addr = 0.0.0.0

# Sensor name use to identify this honeypot instance. Used by the database
# logging modules such as mysql.
#
# If not specified, the logging modules will instead use the IP address of the
# connection as the sensor name.
#
# (default: not specified)
#sensor_name=myhostname

# Fake address displayed as the address of the incoming connection.
# This doesn't affect logging, and is only used by honeypot commands such as
# 'w' and 'last'
#
# If not specified, the actual IP address is displayed instead (default
# behaviour).
#
# (default: not specified)
#fake_addr = 192.168.66.254

# Banner file to be displayed before the first login attempt.
#
# (default: not specified)
#banner_file =

# Session management interface.
#
# This is a telnet based service that can be used to interact with active
# sessions. Disabled by default.
#
# (default: false)
interact_enabled = false
# (default: 5123)
interact_port = 5123

# MySQL logging module
#
# Database structure for this module is supplied in doc/sql/mysql.sql
#
# To enable this module, remove the comments below, including the
# [database_mysql] line.

[database_mysql]
host = localhost
database = kippo
username = root
password = honeydrive
```

10.1.8. Userdb.txt

```
root:0:123456
admin:1:adm1n
```

10.1. Annex 2: Dades recol·lectades durant el període d'escolta

Data	KIPPO	Dioanea	Nºconnexions total per dia	HTTP	SMB	NetBIOS	MsSQL	Sip	TELNET	MySQL	Control TCP	RDP
05-may	3702	10117	13819	13	6487	3167	21	270	25	1	0	18
06-may	3559	3480	7039	11	2636	463	24	219	25	96	0	18
07-may	2952	34099	37051	18	5201	523	54	300	19	42	0	17
08-may	1427	8625	10052	59	21610	11847	38	398	22	2	0	27
09-may	2516	33532	36048	43	5814	2389	14	205	16	3	0	20
10-may	1564	76540	78104	23	23176	9777	88	236	19	36	1	10
11-may	1576	28074	29650	85	8253	2482	28	185	21	61	1	20
12-may	6208	25860	32068	53	20671	6985	58	115	33	102	0	15
13-may	2338	26867	29205	58	17704	6269	52	312	22	2	0	26
14-may	1807	44490	46297	45	17394	8508	59	349	15	52	0	14
15-may	9460	48665	58125	131	28499	15480	18	235	18	6	0	10
16-may	2638	39382	42020	561	32789	14772	15	378	28	6	0	10
17-may	2552	17525	20077	472	23127	12505	29	1233	22	16	0	12
18-may	3240	27693	30933	24	11760	3848	45	1632	16	86	0	29
19-may	2183	14628	16811	40	24554	1298	1	1573	9	126	0	15
20-may	2320	18097	20417	33	12683	688	120	954	18	2	66	7
TOTAL	**50042**	**457674**	**507716**	**1669**	**262358**	**101001**	**664**	**8594**	**328**	**639**	**68**	**268**

10 Serveis més connexions	Nº connexions
SMB	664
NetBIOS	8594
SSH	507716
Sip	639
HTTP	101001
MsSQL	328
MySQL	268
Telnet	68
RDP	0
Control TCP	0

Servei	Nº Atacs
SMB	242758
NetBIOS	0
SSH	50042
Sip	0
HTTP	1420
MsSQL	510
MySQL	480
Telnet	0
RDP	0
Control TCP	0

11. Bibliografia

En aquest apartat es llista la bibliografia feta servir per realitzar aquest projecte.

Wikipedia "Honeypot (computing)"
[en línia] 2014
<http://en.wikipedia.org/wiki/Honeypot_%28computing%29>

Bruteforce "HoneyDrive"
[en línia] 2014
<http://bruteforce.gr/honeydrive>

Security At Work "Honeydrive episode 1"
[en línia] 2014
<http://www.securityartwork.es/2014/03/05/honeydrive-episode-i/>

Segu.info "Boletín 192 Cómo crear un Honeypot paso a paso"
[en línea] 28 d'abril de 2013
<http://www.segu-info.com.ar/boletin/boletin-192-130728.htm>

Carnivore "Dionaea"
[en línia] 2014
<http://dionaea.carnivore.it/>

Bruteforce "Starting with Dionaea malware honeypot"
[en línia] 2014
<http://bruteforce.gr/starting-with-dionaea-malware-honeypot.html>

Honeyd "General Information"
[en línia] 2014
<http://www.honeyd.org/general.php >

Wikipedia "Honeyd"
[en línia] 2014
<http://en.wikipedia.org/wiki/Honeyd>

Infosanity's Blog "Starting with Amun"
[en línea] 15 de maig de 2010
<http://blog.infosanity.co.uk/2010/05/15/starting-with-amun/>

Jan Göbel, "Amun: A Python Honeypot"
[en línea] 2014
<http://pi1.informatik.uni-mannheim.de/filepool/publications/amunhoneypot.pdf>

Security by default, "Honeypot con Kippo - I"
[en línia] 6 de novembre de 2012
<http://www.securitybydefault.com/2012/11/honeypot-con-kippo-i.html>

Bruteforce “Kippo-Graph”
[en línia] 2014
<http://bruteforce.gr/kippo-graph>

VMWare “vCloud Director User’s Guide vCloud director 5.1”
[en línea] 2014
http://pubs.vmware.com/vcd-51/topic/com.vmware.ICbase/PDF/vcd_51_users_guide.pdf