Honeypots

Un "honeypot" és un sistema que simula ser un objectiu real, el qual s'espera que sigui atacat a mode d'"esquer". Els principals objectius són detectar i alertar sobre un atac abans de què afecti a altres sistemes crítics (distraient eventualment l'atacant) però, sobre tot, obtenir informació sobre l'atac i el propi atacant i, d'aquesta manera, descobrir campanyes malicioses i/o noves vulnerabilitats, i aprendre'n. Moltes vegades es combina el seu ús amb algun NIDS per complementar la informació recollida.

Registrant l'activitat de l'atacant (i guardant-la en algun tipus de base de dades per un posterior anàlisi), un "honeypot" pot saber el tipus d'atac utilitzat (és a dir: les comandes emprades, les mostres dels binaris/exploits/malware descarregats/compilats/modificats, les captures de tràfic de xarxa rellevants, els ports accedits, els intents d'autenticació i les contrasenyes emprades, les hores i dates amb més densitat de tràfic, etc) així com la direcció IP de l'atacant -i, per tant, la seva geolocalització, els números ASN associats i la seva reputació segons llistes públiques com les de **VirusTotal, HoneyDB, OpenBL, NoThink, AlientVault**, etc-, el seu sistema operatiu, les eines i metodologia utilitzades, la seva direcció MAC -si fos un atac local-, etc.

> NOTA: Cal tenir en compte que moltes vegades, l'"atacant" resulta ser al seu torn una màquina infectada, de forma que qui està efectual l'atac en realitat és un "exèrcit" de "bots"

Els "honeypots" es disfressen de serveis de xarxa vulnerables capaços d'interactuar amb l'atacant com si fossin serveis reals, responent-hi amb els missatge esperables segons el tipus de protocol de nivell 7 utilitzat en cada cas (DHCP, DNS, HTTP, SSH, etc) emprat pel "honeypot" en qüestió. Fins i tot alguns "honeypots" (com els que simulen un servei FTP, Samba, etc) ofereixen un sistema de fitxers "fals" per tal de què l'atacant hi pugui operar. No obstant, quan un atacant es connecta al servei i tracta de penetrar en ell, encara que el "honeypot" simuli un forat de seguretat, realment no permet guanyar el control del sistema: tot el que l'atacant veu no és real.

La idea és que un "honeypot" no sigui cap sistema de producció; per tant, ningú hauria de tractar de comunicar-se amb ell. Qualsevol tràfic amb destí el "honeypot" hauria de ser considerat sospitós i qualsevol tràfic originat des del "honeypot" significarà un sistema compromès. Moltes vegades el que es fa és deixar a "la intempèrie d'Internet" un "honeypot" aïllat funcionant sobre un servidor públic com ara el que ofereixen empreses amb una cartera de productes "IaaS" com són Linode, DigitalOcean, Vultr, Scaleway, Upcloud o AWS (entre moltes d'altres) i observar el seu comportament però també podem trobar "honeypots" a universitats o sistemes de control industrial, entre d'altres. També és habitual implementar una xarxa de "honeypots" (l'anomenada "honeynet") que romanguin en primera "línia de foc" (els anomenats "sensors"), els quals reben tots els atacs i reenvien -de forma segura- tota la informació recopilada a un servidor central (un SIEM) que l'emmagatzemarà, analitzarà i servirà per consultar-la i visualitzar-la.

Alguns dels projectes de tipus "honeypot" que podem destacar són:

* **Cowrie** (https://github.com/cowrie/cowrie) : "Honeypot" de tipus SSH (i Telnet) dissenyat per registrar intents d'atacs per força bruta i la interacció amb el shell efectuada per l'atacant. Entre les funcionalitats que ofereix està la de simular un sistema de fitxers Debian 5 on es poden afegir o esborrar fitxers i la de guardar els fitxers descarregats per l'atacant amb *wget/curl* (o pujats amb *SFTP/scp*) per tal de poder inspeccionar-los en qualsevol moment.

  > NOTA: Se sol classificar Cowrie com un "honeypot" de tipus "medium interaction" perquè l'atacant pot crear fitxers, descarregar-ne d'altres, observa el seu contingut, etc dins de l'entorn simulat (és a dir, pot realitzar diferents accions). En canvi els "honeypots" de "low" ofereixen molt poc grau d'interacció a l'atacant: només simulen ser un tipus de servei per registrar els intents de connexió i/o inici de sessió i prou (només recopilant les dades pertinents: IP d'origen, usuari i contrasenyes provades, etc)...no simulen cap entorn realista i no permeten a l'atacant fer "res" però serveixen com a indicador d'alerta. A l'altre extrem hi ha els "honeypots" d'alta interacció, els quals sí que ofereixen un entorn real a l'atacant per a què hi pugui desenvolupar la seva activitat de la forma més detallada possible. Cal dir, en aquest sentit, que Cowrie té un mode de funcionament de tipus "proxy" que permet redireccionar les peticions rebudes de l'atacant, un cop registrada la informació pertinent, a un servidor SSH real (en lloc de respondre-les ell mateix mitjançant el seu entorn "fake"), així que en aquest mode podríem considerar-lo com un "honeypot" d'alta interacció.
  > NOTA: Un "honeypot" de tipus SSH de baixa interacció és https://github.com/mushorg/glutton Altres, també de tipus SSH però d'int. mitjana són https://github.com/droberson/ssh-honeypot o https://github.com/jaksi/sshesame

*__Snare__ (https://github.com/mushorg/snare) : "Honeypot" de tipus HTTP/S que detecta les diferents consultes que rep dels eventuals clients (els quals poden no ser "innocents" en el sentit de què aquests podrien ser escanejadors devulnerabilitats) i les mostra a la pantalla. Opcionalment, si aquests events es volen guardar per analitzar-los posteriorment, es pot posar en marxa un programa complementari anomenat **Tanner** (https://github.com/mushorg/tanner), el qual permet avaluar les peticions HTTP i fer-ne una classificació, a més de definir, si s'escau, una determinada resposta al client per a què Snare l'hi envïi i així poder ampliar "el poder d'atracció" per l'atacant.

NOTA: Un altre "honeypot" de tipus HTTP/S és **Galah** (https://github.com/0x4D31/galah) el qual fa ús del LLM OpenAI (és necessita una "API key", doncs) per tal de generar respostes el més úniques i personalitzades possibles segons el context, en lloc d'haver-les de tenir "hardcodeades" al codi

*__Mailoney__ (https://github.com/awhitehatter/mailoney) : "Honeypot" SMTP/S que es pot configurar per ser de baixa, mitjana o alta interacció; molt interessant per estudiar el comportament dels "spammers"

*__Heralding__ (https://github.com/johnnykv/heralding) : "Honeypot" multiprotocol (SSH, HTTP, MySQL, FTP,...) d'interacció baixa que registra intents de connexió.

NOTA: Un altre "honeypot" multiprotocol (però en aquest cas d'interacció mitjana) que és capaç de registrar interaccions i events més enllà dels intents de connexió és **InetSim** (https://www.inetsim.org) però porta temps sense actualitzacions. Altres alternatives multiprotocol són **Fapro** (https://github.com/fofapro/fapro), **Chameleon** (https://github.com/qeeqbox/chameleon i https://github.com/qeeqbox/honeypots) o **OpenCanary** (https://github.com/thinkst/opencanary)

*__Dionaea__ (https://github.com/DinoTools/dionaea): "Honeypot" més especialitzat en la recopilació, execució controlada i anàlisi de malware; aquest programa és recomanable fer-lo servir en paral.lel a l'estudi a l'aula dels diferents tipus de malware existents. Porta temps sense actualitzacions.

NOTA: Teniu una llista molt llarga de diferents tipus de "honeypots" a https://github.com/paralax/awesome-honeypots . D'altra banda, teniu més informació sobre els conceptes, tipus, usos i software relacionat amb els "honeypots" a https://blog.elhacker.net/2021/01/los-mejores-honeypots-ejemplos-y-tipos-trampas-rdp-ssh-cowrie-docker-rdpy.html , així com també al document "Guía de implantación de un honeypot industrial" d'INCIBE (https://www.incibe.es)

EXERCICIS:

1.-En aquest exercici implementarem un "honeypot de joguina" que recollirà en un missatge de log cada intent de connexió TCP diferent realitzat contra un port qualsevol de la nostra màquina (dins d'un determinat rang, que pot ser del 10 al 1000, per exemple). És a dir, implementarem un "honeypot" multiprotocol de nivell 7 (tot i que seria millor dir "multiport") de baixa interacció. Per fer-ho, arrenca una màquina virtual VirtualBox amb un sistema qualsevol però amb la seva tarja de xarxa en mode "adaptador pont".

a) Escriu-hi el següent codi en un arxiu que anomenarem *tinypot.sh* i tot seguit dona-li permisos d'execució (així: *chmod +x tinypot.sh*)::

```
#!/bin/bash
nft add table inet nat
nft add chain inet nat prerouting { type nat hook prerouting priority -100 \; }
nft add rule inet nat prerouting tcp dport { 10-1000 } log prefix \"ATENCIO: \" redirect to 4444
```

NOTA: Aquest codi estableix les regles del tallafocs necessàries d'una banda per redirigir tot el tràfic de xarxa entrant destinat a qualsevol dels ports entre el 10 i el 1000 a un únic port de destí, el 4444 (on estarà escoltant, de seguida ho veurem, el nostre "honeypot") i, de l'altra, guardar al Journal certa informació recollida d'aquest tràfic (com pot ser la IP d'origen o el port de destí, entre altres) i així poder-lo consultar a posteriori

b) Escriu-hi el següent codi en un arxiu que anomenarem *ncpot.sh* i tot seguit dona-li permisos d'execució (així: *chmod +x ncpot.sh*):

```
#!/bin/bash
while [ 1 ]; do nc -l -p 4444 >> /tmp/data.log; done
```

NOTA: Aquest codi posa en marxa el servei pròpiament dit que rebrà les peticions, les dades rebudes de les quals es guardaran en un fitxer. Es podria dissenyar, però, un script més sofisticat que respongués a les peticions i que fins i tot les respongués de forma personalitzada segons el tipus de petició rebuda (si SSH, HTTP, etc) però això ho deixem com a exercici extra

c) Executa l'script "tinypot.sh" així (*sudo ./tinypot.sh*) per activar les regles del tallafocs i tot seguit executa l'script "ncpot.sh" en segon pla així (*systemd-run --user ./ncpot.sh*) per posar en marxa el servidor "fake"

d) Des de la teva màquina real intenta connectar-t'hi a qualsevol port entre 10 i 1000 de la màquina on s'estan executant els scripts anteriors usant diversos programes, com ara el client *nc*, *curl*, etc. Per exemple, així:

nc ip.serv.honey 123 (i tot seguit escriu-hi alguna cosa i tanca la connexió)
curl -d "PEPE" ip.serv.honey 456 (i tot seguit tanca la connexió)
Etc

NOTA: Les comandes anteriors no es poden executar a la mateixa màquina on estàs executant els scripts perquè la cadena PREROUTING del tallafocs no és travessada pel tràfic local a una màquina

e) Atura l'execució i l'efecte dels dos scripts anteriors amb les comandes *sudo pkill ".*pot.sh"* i *sudo nft flush ruleset*. Tot seguit observa el contigut del Journal executant *journalctl -e -k -g ATENCIO*: hauràs de veure els missatges generats pel tallafocs corresponents als intents de connexió realitzats a l'apartat anterior. ¿Quina informació et donen, per exemple, els camps *IN=* , *SRC=* , *DPT=* o *PROTO=* de cada línia de registre?

eII) Observa també el contingut del fitxer "/tmp/data.log" (també amb un simple cat). ¿Què veus?

2.-a) Executa les següents comandes per instal.lar Cowrie a una màquina virtual VirtualBox qualsevol que tinguis disponible (Ubuntu o Fedora) que tingui la seva tarja de xarxa en mode adaptador pont.

A Ubuntu
sudo apt install git libssl-dev libffi-dev build-essential libpython3-dev python3 python3-pip
git clone https://github.com/cowrie/cowrie
cd cowrie && pip3 install --user --upgrade -r requirements.txt
echo export PATH=$PATH:$HOME/.local/bin >> ~/.bashrc

A Fedora
sudo dnf install git openssl-devel libffi-devel @development-tools python3-devel python3 python3-pip
git clone https://github.com/cowrie/cowrie
cd cowrie && pip3 install --user --upgrade -r requirements.txt
Opcional: *sudo systemctl stop firewalld && sudo systemctl disable firewalld*

b) Canvia el nom de l'arxiu "userdb.example" (ubicat a la carpeta "/home/usuari/cowrie/etc") per "userdb.txt" i dedueix, a partir dels comentaris que apareixen al seu contingut, quins usuaris i contrasenyes podrem fer servir per defecte a l'hora d'iniciar sessió en un eventual servidor Cowrie de tipus SSH.

c) Dedueix, a partir dels comentaris que apareixen en el contingut de l'arxiu "cowrie.cfg.dist" (ubicat a la carpeta "/home/usuari/cowrie/etc"), per a què serveixen les següents línies de configuració (no en modifiquis cap!), d'entre totes les que hi apareixen, i pren nota del seu valor per defecte que tenen:

NOTA: La configuració per defecte de Cowrie s'emmagatzema a l'arxiu "cowrie.cfg.dist" però és molt recomanable no modificar-lo perquè es podria sobreescriure automàticament amb les actualitzacions del programa. Si es vol canviar el valor de directives concretes, aquestes es poden indicar en un altre arxiu anomenat "cowrie.cfg" (i ubicat a la mateixa carpeta), el qual no es tocarà mai quan hi hagin actualitzacions. Tots dos fitxers es llegeixen a l'inici, i les entrades repetides que apareixen a "cowrie.cfg" tenen prioritat a les que apareixen a "conwrie.cfg.dist".

* Dins de la secció "*[honeypot]*":

hostname
download_path ("artifact" vol dir binari,malware...tot allò que l'atacant pugui descarregar-s'hi allà)

contents_path (per a què la modificació del sistema de fitxers sigui efectiva, però, a més de copiar-hi allà els fitxes desitjats, caldrà emprar tot seguit la comanda *bin/fsctl* o *bin/createfs* per afegir les metadades adients -nom, ruta, permisos, propietari, mida,...- de cara a comandes com *ls*, etc dins un fitxer especial anomenat "cowrie/share/cowrie/fs.pickle". Ho veurem)
txtcmds_path (la carpeta en qüestió es troba sota carpeta "cowrie/share/cowrie")
interactive_timeout i *authentication_timeout*

* Dins de la secció "*[shell]*":

filesystem
processes
arch i *kernel_version* i *ssh_version*

* Dins de la secció "*[ssh]*":

enabled
listen_endpoints (observar el valor "tcp:2222:interface=0.0.0.0")
sftp_enabled

* Dins de la secció "*[output_jsonlog]*":

enabled
logfile

NOTA: Altres sortides a més d'arxius JSON, són, per exemple, servidors OpenSearch, MySQL, SQLite, MongoDB, RethinkDB, Redis, InfluxDB, Splunk, Kafka, entre molts d'altres més

d) Posa en marxa Cowrie executant dins de la carpeta "cowrie" (i com usuari normal) la comanda *bin/cowrie start* Des de la màquina real executa la comanda *ssh -p 2222 oracle@ip.maq.virt* i indica qualsevol contrasenya ¿Què passa? ¿Què veus en executar *ls /bin*? ¿I en executar *ls /home*? ¿Què veus en executar *cat /etc/passwd*? ¿I *cat /etc/issue*? Surt de l'entorn simular amb *exit* i atura Cowrie amb la comanda *bin/cowrie stop*

L'arxiu "/etc/passwd" proporcionat per defecte per l'entorn "simulat" de Cowrie llista un usuari anomenat "phil". Un atacant que conegui Cowrie això ho sap i podria fàcilment descobrir que està en un entorn "simulat" si veu que existeix aquest usuari allà present. Una manera de mitigar aquest problema és canviar-li el nom a l'usuari per a què sigui un altre.

e) Un cop dins de la carpeta "cowrie", edita l'arxiu "honeyfs/etc/passwd" per tal de canviar el nom "phil" per un altre (per exemple, "joe"). Tot seguit, executa la comanda *python3 bin/fsctl share/cowrie/fs.pickle* i, a l'entorn interactiu que apareix (del qual pots obtenir una llista de les comandes disponibles fent *help*), executa *mv /home/phil /home/joe* i surt. Torna a repetir l'apartat anterior i ara comprova que, efectivament, l'usuari "joe" és ara el que apareix llistat a l'arxiu "/etc/passwd" i que existeix la seva corresponent carpeta personal. Atura de nou Cowrie

Si volem afegir/modificar/esborrar qualsevol fitxer o carpeta del sistema de fitxers simulat per a què sigui de forma permanent (ja que els canvis fets "en calent" per un usuari loguejat a Cowrie es perden sempre en sortir de la sessió), a més d'editar el contingut pròpiament dit de la carpeta "honeyfs", hem d'utilitzar també la comanda *fsctl* per editar l'arxiu "fs.pickle", que és on es guarden totes les referències. Concretament, a l'apartat següent afegirem un nou fitxer a la carpeta personal de l'usuari "joe"

f) Un cop dins de la carpeta "cowrie", executa *echo "Hola amic" > honeyfs/home/joe/flag.txt* (potser cal crear primer les carpetes "honeyfs/home" i "honeyfs/home/joe") Tot seguit, executa la comanda *python3 bin/fsctl share/cowrie/fs.pickle* i, a l'entorn interactiu que apareix, executa...:

touch /home/joe/flag.txt 1234 (el número representa la mida en bytes que "tindrà" el fitxer)
chown 1000 /home/joe/flag.txt (només s'admet indicar UIDs, no noms)
chgrp 1000 /home/joe/flag.txt (només s'admet indicar UIDs, no noms)
exit

... i arrenca de nou Cowrie per tal d'inicia-hi una sessió SSH de nou amb l'usuari "oracle"; un cop a dins, què mostra per pantalla la comanda *cat /home/joe/flag.txt* ?

g) Per fer que Cowrie es posi en marca com un dimoni Systemd "estàndar", un cop dins de la carpeta "cowrie" cal que facis el següent:

*.- Crea a la carpeta "/etc/systemd/system" un fitxer de nom "cowrie.socket" amb aquest contingut:

[Unit]
Description=Cowrie SSH and Telnet honeypot socket
[Socket]
ListenStream=0.0.0.0:2222
#Si es vol activar també el honeypot Telnet caldrà afegir una altra línia així: ListenStream=0.0.0.0:2223
[Install]
WantedBy=sockets.target

*.- Crea a la carpeta "/etc/systemd/system" un fitxer de nom "cowrie.service" amb aquest contingut (adapta'l al nom i ruta de la carpeta personal del teu usuari en particular):

[Unit]
Description=A SSH and Telnet honeypot service
After=network.target
Requires=cowrie.socket
[Service]
Type=simple
User=usuari
Group=usuari
Environment=PATH=/bin:/usr/bin:/home/usuari/.local/bin
Environment=COWRIE_STDOUT=yes
ExecStart=/home/usuari/cowrie/bin/cowrie start
Restart=always
[Install]
WantedBy=multi-user.target

NOTA: La variable COWRIE_STDOUT serveix per no posar en marxa Cowrie en segon pla, ja que d'això es vol que s'encarregui Systemd (de fet, per això s'ha indicat que el servei és de tipus "simple"); a més, com diu el seu nom, envia els logs a la sortida estàndar (i això en un dimoni Systemd significa que en realitat s'enviarà al Journald). La variable PATH cal explicitar-la per a què l'script Python cowrie trobi els seus components i mòduls correctament (sobre tot l'script "twistd" en el què es basa)

*.- Crea l'arxiu "etc/cowrie.cfg" amb el següent contingut (això és per fer que sigui el socket Systemd qui s'encarregui d'escoltar al port indicat i no Cowrie directament com indica el valor d'aquesta directiva indicat a l'arxiu "cowrie.cfg.dist"):

[ssh]
listen_endpoints = systemd:domain=INET:index=0

NOTA: Per cada línia *Listen Stream=* que aparegui a l'arxiu *.socket caldrà afegir dins de l'arxiu "cowrie.cfg" una línia *listen_endpoints* amb un valor de "index" correlatiu

*.- Executa la comanda: *sudo systemctl --now enable cowrie.socket*

h) Des de la màquina real torna a executar la comanda *ssh -p 2222 oracle@ip.maq.virt* No hauries de veure cap diferència respecte el que vas veure a l'apartat d).

El fet de posar en marxa Cowrie com usuari normal provoca que aquest programa no pugui associar-se al port 22 (que és el port per defecte on escolta un servidor SSH estàndar, com ja sabeu) ja que només els serveis iniciats com root poden associar-se a ports menors del 1024. Hi ha dues solucions per posar en marxa Cowrie amb un usuari normal i que els clients SSH no hagin de connectar-se a un port diferent de l'estàndar:

1.- Assignar la capability CAP_NET_BIND_SERVICE a l'intèrpret Python, així: *sudo setcap cap_net_bind_service=+ep /usr/bin/python3* D'aquesta manera, Cowrie podrà escoltar a qualsevol port encara que l'hagi iniciat un usuari sense privilegis. El problema és que qualsevol altre programa interpretat per Python també, cosa que podria ser un problema, i és per això que es recomana la segona opció.

2.- Redireccionar mitjançant el tallafocs del sistema Nftables tot el tràfic que arribi de part dels clients al port 22 fins el port 2222, on estarà escoltant Cowrie com sempre. Això es pot aconseguir simplement afegint les següents regles:

```
sudo nft add table inet nat
sudo nft add chain inet nat prerouting { type nat hook prerouting priority -100 \; }
sudo nft add rule inet nat prerouting tcp dport 22 redirect to 2222
```

i) Segueix les instruccions del pas nº2 indicat als paràgrafs blaus anteriors (tenint en compte de no tenir iniciat cap servidor SSH "estàndard") i seguidament, des de la màquina real executa ara la comanda *ssh oracle@ip.maq.virt* No hauries de veure cap diferència respecte el que vas veure a l'apartat d).

NOTA: Una altra comanda interessant de provar des d'una màquina client seria *nmap -A ip.maq.cowrie*

j) Instal.la, si no ho està ja, el paquet "jq" a la màquina on s'estigui executant Cowrie i seguidament, executa la comanda *jq "." /home/usuari/cowrie/var/log/cowrie/cowrie.json | less* . ¿Què veus? ¿Quins events representen els eventids "cowrie.session.connect", "cowrie.login.success", "cowrie.login.failed", "cowrie.command.input" o "cowrie.session.close"?

k) ¿Què veus (suposant que has arrencat Cowrie algun cop en forma de servei Systemd), si executes la comanda *journalctl -e -u cowrie -g "CMD:"* ? ¿ I si executes *journalctl -e -u cowrie -g "login attempt" ?*

NOTA: Si Cowrie no ha sigut executat via Systemd, llavors el log anterior s'emmagatzemarà, enlloc de al Journal, en l'arxiu "/home/usuari/cowrie/var/log/cowrie/cowrie.log" (tal com s'indica, de fet, a la configuració indicada "cowrie.cfg.dist"

l) ¿Què passa si executes (des de dins de la carpeta "cowrie") aquesta comanda: *python3 bin/playlog var/lib/cowrie/tty/fitxerambnomrandom* ? ¿Què hi tenen a veure les línies *ttylog=* i *ttylog_path=* del fitxer de configuració "cowrie.cfg.dist"?

m) Digues, en un paràgraf, què explica aquest article https://cowrie.readthedocs.io/en/latest/PROXY.html ¿Per a què serveixen, llavors, les línies *backend=proxy* de la secció "*[honeypot]*" i *backend=simple*, *backend_ssh_host=localhost* i *backend_ssh_port=2022* de la secció "*[proxy]*" del fitxer de configuració "cowrie.cfg.dist"?

2BIS.-Una idea molt interessant seria enviar les dades recollides per Cowrie a un servidor OpenSearch (via FluentBit, per exemple) per tal de, amb l'ajut de Dashboards, obtenir gràfics "de formatge" classificant els atacs per IP d'origen (amb geolocalització), o rangs de temps, o tipus de servidor atacat (si tinguéssim també el "honeypot" Telnet funcionant, etc). Fem-ho:

a) Instal·la i arrenca, si no ho tens fet ja, els servidors OpenSearch/Dashboards tal com s'indica en la teoria corresponent (és a dir, via Podman).

b) Instal·la, si no ho tens fet ja, el servidor Fluent-Bit tal com s'indica en la teoria corresponent. Tot seguit, edita l'arxiu de configuració "/usr/local/etc/fluent-bit/fluent-bit.conf" (que és l'utilitzat per defecte pel servei "fluent-bit.service") per a què només tingui aquest contingut:

```
[INPUT]
  Name tail
  Path /home/usuari/cowrie/var/log/cowrie/cowrie.json
```

[OUTPUT]
Name opensearch
*Match **
Host 127.0.0.1
Port 9200
Index cowrie
HTTP_User admin
*HTTP_Passwd Un4C0ntraSs3ny4C0mplicad4_*
Tls on
Tls.verify off
Suppress_Type_Name On

... i tot seguit inicia el servei Fluent-bit (*sudo systemctl start fluent-bit*)

c) Després de crear al Dashboars un "index pattern" anomenat "cowrie", hauries de començar a rebre a la seva pantalla "Discover" els logs de Cowrie. A partir d'observar els camps concrets rebuts a cada missatge, realitza llavors els següents diagrames:

*"Pie chart" mostrant el nº d'intents d'inici de sessió exitosos classificats per IP d'origen
*"Pie chart" mostrant el nº d'intents d'inici de sessió fallits classificats per IP d'origen
* "Line" mostrant la mateixa informació que els dos "pie chart" anteriors però al llarg del temps

NOTA: Afegint-hi geolocalització (això es podria fer mitjançant filtres de FluentBit o també d'OpenSearch), en lloc de classificar per IP d'origen es podria fer per país d'origen, i així mostrar aquesta informació en un mapa de Dashboards. Ho deixem com a exercici voluntari

*"Pie chart" comptant el nº de cops que s'ha intentat iniciar sessió amb un determinat nom d'usuari
*"Table" mostrant la mateixa informació que el "pie chart" anterior
* "Vertical bar" mostrant la mateixa informació que el "pie chart" anterior però al llarg del temps
*"Pie chart" comptant els noms d'usuaris intentats per IP d'origen (és un "pie chart" de doble corona)
*"Pie chart" comptant el nº de cops que s'ha intentat iniciar sessió amb una determinada contrasenya
*"Table" mostrant la mateixa informació que el "pie chart" anterior
*"Pie chart" comptant el nº de cops que l'atacant, un cop loguejat, ha executat cada comanda
*"Table" mostrant la mateixa informació que el "pie chart" anterior
* "Vertical bar" mostrant la mateixa informació que el "pie chart" anterior però al llarg del temps
*"Pie chart" comptant les comandes executades per noms d'usuari (és un "pie chart" de doble corona)

NOTA: Si no es vol gestionar tota la recollida, processament i visualització de dades del nostre honeypot, existeix la possibilitat d'instal·lar-hi un petit "agent" que enviarà aquestes dades a un servidor remot online que farà tota aquesta feina per nosaltres. A més, d'aquesta manera podreu decidir també si voleu compartir aquesta informació recollida amb altres operadors de "honeypots", per tenir-ne una base de dades d'atacs més àmplia i completa Teniu més informació de com implementar aquest sistema a https://honeydb.io/deploy

3.-a) Instal.la el "honeypot" multiprotocol Heralding seguint els següents passos:

A Ubuntu
sudo apt install git libssl-dev libffi-dev build-essential libpython3-dev python3 python3-pip
git clone https://github.com/johnnykv/heralding.git
cd heralding && pip3 install --user -r requirements.txt
sudo python3 setup.py install

A Fedora
sudo dnf install git openssl-devel libffi-devel @development-tools python3-devel python3 python3-pip
git clone https://github.com/johnnykv/heralding.git
cd heralding && pip3 install –user -r requirements.txt
sudo python3 setup.py install

b) Posa'l en marxa seguint els següents passos (has de tenir en compte de no estar utilitzant els ports per on Heralding escoltarà, com són els estàndards 21, 22, 23, 25, 80, 110, 143, 443, 465, 993 o 995, o altres com per exemple el nº 1080 -de Socks5-, o el nº 3306 o nº 5432 -de MySQL o PostgreSQL, respectivament- o el nº 3389 o nº 5900 -de l'escriptori remot RDP o VNC, respectivament-, etc):

mkdir tmp && cd tmp
sudo heralding

c) Intenta connectar via SSH, via HTTP/S (amb un navegador) i també via VNC (amb el client Remmina) des de la màquina real fent servir usuaris i contrasenyes a l'atzar. Tot seguit para Heralding i observa el contingut dels arxius "log_auth.csv" i "log_session.csv" ubicats dins la carpeta "tmp". ¿Quina informació hi ha guardada allà? A partir del vist en els arxius anteriors i del comportament observat del programa, ¿és Heralding un "honeypot" de tipus "low interaction" o de tipus "high interaction"?

4.-a)¿Què ofereix aquest projecte: https://github.com/telekom-security/tpotce ?

> NOTA: Tens més informació sobre les capacitats i funcionalitats d'aquest projecte en els articles https://blog.elhacker.net/2021/01/instalar-honeypot-t-pot-en-una-maquina-virtual-tpotce-cowrie-docker-dionea.html i https://pixuetuvalley.com/que-es-y-como-se-monta-una-honeypot-con-tpot i també en https://sicherheitstacho.eu

b) Descarrega't la ISO més moderna de tipus "amd64" disponible al Github de TpotCE i instal·la-la en una màquina virtual VirtualBox nova amb la seva tarja de xarxa en mode adaptador pont. El procés d'instal·lació serà similar al d'una instal·lació d'un sistema Debian estàndard; un cop acabi la instal·lació, en reiniciar des del disc dur apareixerà un quadre on es preguntarà el tipus de "honeypot" que es desitja (respón "Standard"), a més de la contrasenya desitjada per accedir al terminal (l'usuari per defecte és "tsec") i també l'usuari/contrasenya desitjat per accedir al panell web, al qual, un cop finalitzat tot aquest procés, s'hi podrà accedir amb un navegador anant a l'adreça https://ip.maq.virt:64297 Entra-hi: ¿què veus?

> NOTA: Existeix la possibilitat d'instal·lar TpotCE en un sistema ja instal·lat, però aquest sistema ha de ser igualment Debian, si no la instal·lació no funciona

Un "token canari" és un enllaç únic dissenyat per detectar quan algú hi fa clic, el comparteix o interactua amb ell d'alguna manera. Pots pensar-lo com un "cable de seguretat" deixat pels defensors per avisar-los quan algú està mirant per algun lloc de la teva xarxa on no hi hauria d'estar.

> NOTA: Degut a la manera com moltes aplicacions obtenen una vista prèvia d'URL per als enllaços compartits en xats privats, els tokens canaris fins i tot poden "trucar a casa" quan algú consulta un xat privat sense fer clic a l'enllaç

5.-a) Vés a https://canarytokens.org/generate (el codi font del qual, per si es vol implementar un servei similar de forma "self-hosted", es troba a https://github.com/thinkst/canarytokens) i, dins del desplegable "Select your token", tria l'opció de "Web bug/URL token"; indica a més la teva adreça de correu i un text que representarà el missatge d'avís. El resultat serà una URL que hauràs d'incloure allà on vulguis (en el codi d'una pàgina web que hauria de ser poc accessible, o en el contingut d'un email o document que no hauria d'esser massa difós, etc) per ésser notificat que algú ho hagi llegit. Per fer la prova, clica-hi tu mateix directament en la URL obtinguda i tot seguit comprova si t'ha arribat un correu d'avís a la teva bústia.

b) Prova altres tipus de "canary token", com per exemple l'anomenat "Adobe Acrobat PDF Document" o també el "DNS token", entre altres.

6.-a) Un tipus molt específic de "honeypots" són els de tipus ICS (també anomenats SCADA). Llegeix el següent article (https://www.tripwire.com/state-of-security/ics-security/ics-security-challenge-organizations) i digues què són. En aquest sentit, ¿què creus que fa aquest programa: https://github.com/mushorg/conpot ?

NOTA: Vocabulari-> Els PLC són elements programables que interaccionen amb actuadors i sensors per tal d'obrir vàlvules, regular valors, fer funcionar motors, engegar bombes, etc. Es poden trobar a tot arreu, des de muntanyes russes fins a taulers de control per a instal·lacions nuclears. Els PLC actuen tant en l'espai digital com en l'espai físic, i això els fa molt interessants per a actors maliciosos, ja que un sistema defectuós podria tenir resultats físics catastròfics.

b) Si s'implementa un "honeypot" en Internet (mitjançant la contractació d'algun proveïdor "IaaS"), tard o d'hora escanejadors de dispositius com **Shodan** (https://www.shodan.io) o **Censys** (https://search.censys.io) el trobaran, i això farà que els atacants siguin conscients de la seva existència fent una simple recerca en aquests serveis. En aquest sentit, ¿per a què creus que serveix aquesta web: https://honeyscore.shodan.io ?