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 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.

EXERCICIS:

Cowrie (https://github.com/cowrie/cowrie) és un "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.

1.-a) Executa les següents comandes per instal.lar Cowrie a una màquina virtual que tinguis disponible (Ubuntu o Fedora) amb una 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 http://github.com/cowrie/cowrie
cd cowrie
python3 -m pip install –user --upgrade -r requirements.txt (o *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 http://github.com/cowrie/cowrie
cd cowrie
python3 -m pip install –user --upgrade -r requirements.txt (o *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 del 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) Canvia el nom de l'arxiu "cowrie.cfg.dist" (ubicat a la carpeta "/home/usuari/cowrie/etc") per "cowrie.cfg". Seguidament dedueix, a partir dels comentaris que apareixen al seu contingut, per a què serveixen les següents línies de configuració (no cal que modifiquis cap, de moment):

sensor_name	(dins de la secció [honeypot])
hostname	(dins de la secció [honeypot])
log_path	(dins [honeypot]; el seu valor és la ruta d'una carpeta; el fitxer corresponent s'anomena cowrie.log)
download_path	(dins de la secció [honeypot]; "artifact" vol dir binari, malware, etc)
contents_path	(dins de la secció [honeypot]; feel free to copy a real system here or use bin/fsctl or bin/createfs)
txtcmds_path	(dins de la secció [honeypot])
ttylog_path	(dins [honeypot]; activat amb la línia ttylog=true; el programa bin/playlog permet revisualitzar-los)
interactive_timeout	(dins de la secció [honeypot])
authentication_timeout	(dins de la secció [honeypot])
auth_class	(dins de la secció [honeypot]; observar la diferència entre el seu valor "UserDB" o "AuthRandom")
filesystem	(dins de la secció [shell])
processes	(dins de la secció [shell])
arch	(dins de la secció [shell])
kernel_version	(dins de la secció [shell])
enabled	(dins de la secció [ssh])
version	(dins de la secció [ssh])
listen_endpoints	(dins de la secció [ssh]; observar el valor "tcp:2222:interface=0.0.0.0")
sftp_enabled	(dins de la secció [ssh])
forward_redirect	(dins de la secció [ssh]; en combinació amb la/s línia/es *forward_redirect_nº = IP:nº*)
enabled	(dins de la secció [output_jsonlog]; en combinació amb la línia *logfile)*
enabled	(dins de la secció [output_elasticsearch]; en combinació amb les línies *host, port* i *index)*

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

NOTA: The configuration for Cowrie is stored in cowrie.cfg.dist and cowrie.cfg. Both files are read on startup, where entries from cowrie.cfg take precedence. The .dist file can be overwritten by upgrades, cowrie.cfg will not be touched

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* ¿Què passa? ¿Què veus en executar *ls /bin*? ¿Què veus en executar cat */etc/passwd*? ¿I *cat /etc/issue*?
e) Per fer que Cowrie es posi en marca com un dimoni Systemd "estàndar", cal que facis el següent:

1.- 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

2.- 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
ExecStart=/home/usuari/cowrie/bin/cowrie start
Restart=always
[Install]
WantedBy=multi-user.target

3.-Adapta l'script Python "/home/usuari/cowrie/bin/cowrie" (o on estigui) per a què funcioni correctament amb el fitxer "cowrie.service" de Systemd creat al punt anterior. Concretament:

*Fes que la línia *DAEMONIZE=""* aparegui com *DAEMONIZE="-n"*
*Fes que la línia *#STDOUT="no"* aparegui com *STDOUT="yes"*
*Afegeix la part indicada en negrita a la línia (canvia el número de versió si cal):
*export PYTHONPATH=${PYTHONPATH}:${COWRIEDIR}/src:**${HOME}/.local/lib/python3.6***
*Escriu sota (o sobre) la línia anterior aquesta nova línia: *export PATH=${PATH}:${HOME}/.local/bin*

NOTA: El primer canvi és 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"). El segon canvi és per enviar els logs a la sortida estàndar, i això en un dimoni Systemd significa que en realitat s'enviarà al Journald. La definició de les variables d'entorn és per a què l'script Python cowrie trobi els seus components i mòduls correctament (sobre tot l'script "twistd" en el què es basa

4.-Comenta la línia *listen_endpoints=tcp:2222:interface=0.0.0.0* de l'arxiu cowrie.cfg i descomenta la línia *listen_endpoints=systemd:domain=INET:index=0* per tal de fer que sigui el socket Systemd que s'encarregui d'escoltar al port indicat i no Cowrie directament.

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

5.-Executa les comandes: *sudo systemctl enable cowrie.socket && sudo systemctl start cowrie.socket*

f) 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 (Iptables/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 una regla com la següent: *sudo iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222*

g) Segueix les instruccions del pas nº2 indicat als paràgrafs blaus anteriors 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*

h) 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.login.succes", "cowrie.login.failed", "cowrie.session.connect", "cowrie.session.close" o "cowrie.command.input" ?

i) Si Cowrie ha sigut executat via Systemd, l'arxiu "/home/usuari/cowrie/var/log/cowrie/cowrie.log" deixa de tenir gaire importància perquè el registre de tot el que passa s'envia ara al Journald. ¿Què veus, per tant, si executes la comanda: *journalctl -e -u cowrie | grep "Command found"*

2.-a) Una idea molt interessant seria enviar les dades recollides per Cowrie a un servidor ElasticSearch (via FileBeat i a través de Logstash) per tal de, amb l'ajut de Kibana, 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). En aquest sentit, ¿quina funció tindria el contingut dels següents fitxers? I més en concret, ¿per a què serveixen els diferents filtres indicats al fitxer de configuració de LogStash ("json", "date", "mutate" (add_field i remove_field), "dns" i "geoip")? ¿I el *if[src_ip]*? Repassa la documentació de LogStash si ho necessites.

/etc/filebeat/filebeat.yml

```
filebeat.inputs:
- type: log
  paths:
    - /home/usuari/cowrie/log/cowrie.json*
output.logstash:
  hosts: ["127.0.0.1:5044"]
```

/etc/logstash/conf.d/logstash.conf

```
input {  beats { port => 5044  } }
filter {
      json {  source => message   }
      date { match => [ "timestamp", "ISO8601" ]  }
      if [src_ip]  {
         mutate { add_field => { "src_host" => "%{src_ip}" }  }
         dns {
            reverse => [ "src_host" ]
```

```
            nameserver => [ "8.8.8.8", "8.8.4.4" ]
            action => "replace"
          }
          geoip {
            source => "src_ip"
            target => "geoip"
          }
        }
        mutate {  remove_field => [ "source", "offset", "input_type" ]  }
}
output {  elasticsearch {  hosts => ["localhost:9200"]  } }
```

b) ¿Què explica aquest article i quina utilitat tindria el que s'hi explica respecte l'ús de la suite ElasticSearch amb Cowrie?: https://www.elastic.co/blog/custom-basemaps-for-region-and-coordinate-maps-in-kibana

c) ¿Què explica aquest article, en general: https://blog.cdemi.io/analyzing-data-from-a-public-facing-honeypot ?

3.-a) Executa les següents comandes a una màquina virtual qualsevol (amb tarja en mode adaptador pont):

```
sudo apt install golang-go
go get github.com/magisterquis/vnclowpot
go install github.com/magisterquis/vnclowpot
cd go/bin
./vnclowpot -l 0.0.0.0:5900
```

A continuació executa el client VNC Remmina de la màquina real ¿Què passa al terminal on s'està executant la comanda "vnclowpot"? ¿Quin tipus de "honeypot" estàs fent servir, doncs?

> NOTA: La distinció entre "low" i "high" ve donada pel grau d'interacció que el "honeypot" ofereix a l'atacant. En aquest sentit, Cowrie és un "honeypot" d'interacció mitjana 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, "vnclowpot" no permet cap tipus d'interacció: només es dedica a recollir els intents de conexió i prou. D'aquí el seu nom.

b) Com pots veure, existeixen diferents "honeypots" segons el protocol de les connexions que es vulguin monitoritzar. En aquest sentit, digues (no cal que l'instal.lis) per a què creus que seria útil executar aquest altre "honeypot": https://github.com/awhitehatter/mailoney

4.-No només podem fer servir "honeypots" especialitzats en un determinat tipus de protocol, però. Existeixen "honeypots" multiprotocol com per exemple https://github.com/johnnykv/heralding . En aquest exercici se't convida a provar-lo. Concretament:

a) Instal.la'l 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:

mkdir tmp && cd tmp
sudo heralding
Veient el que mostra a pantalla un cop iniciat, ¿quins tipus de servidors és capaç de simular Heralding?

c) Intenta connectar via SSH i també via VNC (Remmina) des de la màquina real fent servir contrasenyes a l'atzar. ¿Què veus a la pantalla on s'està executant Heralding?

d) 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?

e) A partir del vist a l'apartat anterior i del comportament observat del programa, ¿és Heralding un "honeypot" de tipus "low interaction" o de tipus "high interaction"?

5.-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, ¿per a què creus que serviria aquest programa: https://github.com/schmalle/medpot ?

b) ¿Per a què serveix aquest programa: https://github.com/huuck/ADBHoney? (interessant llegir el Readme)

c) ¿Per a què serveix aquest programa: https://github.com/mushorg/oschameleon ?

d) ¿Què ofereix aquest projecte: https://github.com/dtag-dev-sec/tpotce ?

e) Visita https://sicherheitstacho.eu i digues què hi veus. ¿Què vol dir "partner's honeypot"?

6.-Els "chisels" de Sysdig també es poden fer servir com a sensors per implementar un "honeypot", tal com s'explica a https://sysdig.com/blog/fishing-for-hackers , a https://sysdig.com/blog/fishing-for-hackers-part-2 i també a https://labs.mwrinfosecurity.com/blog/high-interaction-honeypots-with-sysdig-and-falco Digues en concret, què mostren els "chisels" següents (pots consultar la documentació oficial de Sysdig o repassar els exercicis que vam fer estudiant-lo) i en quin contexte s'utilitzen en els articles anteriors:

topprocs_net
topconns
list_login_shells
topfiles_bytes proc.name contains tar
spy_users proc.loginshellid=1743
echo_fds fd.filename=zbbpxdqalfe.sh

7.-Thug (https://github.com/buffer/thug) és un "honeyclient", és a dir, és un programa (concretament, un script Python) que tal com diu a la seva pàgina web: "*will visit a site and present itself as an exploitable browser (or any of several built-in user-agents) but instead of get compromised, it displays and saves all files that were thrown at it and makes a graph of all the interactions for you. It's very useful to quickly and safely investigate a potentially malicious website*". Un cop sabut això, i després d'haver llegit el següent tutorial (http://909research.com/how-to-use-thug-honeyclient) digues què és el que es mostra a la següent animació: http://909research.com/content/images/2015/07/thug_demo.html

Altres projectes

*Un "honeypot" multiprotocol (SSH, HTTP, MySQL, FTP,...) d'interacció mitjana que és capaç de registrar interaccions i events més enllà dels intents de connexió (com fa "Heralding"): https://www.inetsim.org

*Un "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 de vulnerabilitats) i les mostra a la pantalla: https://github.com/mushorg/snare 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.

*Un "honeypot" de tipus SSH una mica més realista que Cowrie perquè, de fet, utilitza una implementació real de servidor OpenSSH: https://github.com/mushorg/glutton . Una alternativa amb els mateixos objectius és la combinació dels programes https://github.com/amv42/cowrie-sshd i https://github.com/amdv42/sshd-honeypot , els quals treballen en conjunt i es basen en el codi original de Cowrie.

*Un "honeypot" més especialitzat en la recopilació, execució controlada i anàlisi de malware: https://github.com/DinoTools/dionaea Aquest programa és recomanable fer-lo servir en paral.lel a l'estudi a l'aula dels diferents tipus de malware existents.

*Un híbrid entre "honeypot" i eina de hardering: https://github.com/BinaryDefense/artillery

* El projecte https://github.com/threatstream/mhn ofereix la possibilitat d'afegir un panell de control web per tal de gestionar d'una forma més còmoda granjes de honeypots, monitoritzar el seu estat, descarregar els "artifacs" desitjats, etc.

*El projecte https://github.com/honeytrap/honeytrap ofereix una infraestructura de múltiples "honeypots" de tipus sensors que es comuniquen amb altres "honeypots" de tipus receptor centralitzats d'una forma integrada

*Diferents "honeypots" de molts tipus desenvolupats per un mateix senyor: https://github.com/schmalle

*Una llista molt llarga de diferents tipus de "honeypots": https://github.com/paralax/awesome-honeypots

*Una llista de diferents "honeypots" de tipus ICS/SCADA: https://github.com/ITI/ICS-Security-Tools/tree/master/tools/honeypots