Project 3 Final Report - IOT Device Defender

Stephen Bray, Brandon Ko, Imran Hasan

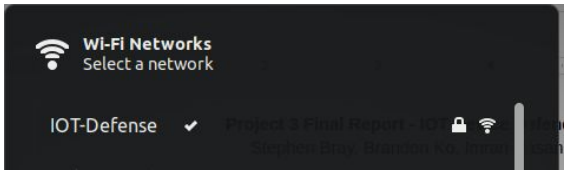
Overview

For our project we decided that we wanted to build a device that could help defend vulnerable IOT devices through metrics and firewall rules. IOT is one of the largest growing industries right now and with that comes a lot of growing pains. One of the main concerns with IOT devices currently is their security. To help address this we decided to not only build something that could monitor these devices but also actively block traffic.

Design

We designed our defender using a raspberry pi 3 as the hardware that would run as our router. By utilizing the onboard wifi chip and the ethernet port, we would turn the pi into a router that would provide internet to any connected device. Then, we would set up a firewall in between the bridge from the wifi interface to the output ethernet interface. This firewall would dynamically sniff traffic and accept or block packets that it deemed fit.

Build



Brandon was in charge of setting up the pi so that it would be usable as a wifi router. To do so, he first installed a new operating system on the SD card and set up ssh. From there he ssh'd into the pi and installed a program called hostapd. Hostapd is a widely used service capable of turning any computer with a wifi chip into a router that other devices can connect to for internet connectivity. He then set up hostapd to forward the traffic coming from wlan0 to eth0 which was connected by ethernet cable to the home router.

From here Imran and Stephen set up a service called NetFilterQueue. NetFilterQueue injects itself into iptable rules and can dynamically accept and drop packets based on further inspection. It also provides a nice and easy to use python interface when paired with a library called Scapy. Using this library, we set up a system to inspect all traffic coming in on wlan0 and keep track of each device connected to our router. We set up a timer for each new device that lasts one day from the time of its first connection. During this one day we track the typical connections that this device tends to make for future reference. After that one day timer, we then lock down all traffic that doesn't match the typical traffic that we tended to see in that original day.

It is in no way perfect, but luckily the system in place is extremely easy to build on top off and can easily be extended. One of our main goals with this project was to build a system that could in the future be used as a base for more projects and extensions. We are excited to keep working on this and hope to have a more complete answer to one of tech's biggest problems.

Firewall Rules and Forwarding

```
iptables: Permission denied (you must be root).
pi@bepis:~ $ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
NFQUEUE    all  --  anywhere             anywhere             NFQUEUE num 1

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
NFQUEUE    all  --  anywhere             anywhere             NFQUEUE num 1

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
pi@bepis:~ $ sudo iptables -t nat
```

Usage

```
pi@bepis:~ $ vim packet-filter.py
pi@bepis:~ $ sudo python packet-filter.py
[*] waiting for data
> New device at ip: 0.0.0.0
>> New port on ip: 0.0.0.0 port: 67
> New device at ip: 169.254.188.160
>> New port on ip: 169.254.188.160 port: 5353
> New device at ip: 192.168.220.64
>> New port on ip: 192.168.220.64 port: 5353
>> New port on ip: 192.168.220.64 port: 53
>> New port on ip: 192.168.220.64 port: 16384
>> New port on ip: 192.168.220.64 port: 16385
>> New port on ip: 192.168.220.64 port: 16386
>> New port on ip: 192.168.220.64 port: 16403
^Cpi@bepis:~ $
```