**1/9/24:**

* Setting up three(3) ubuntu machines on the 2 closest computers to the door at the middle table and the back left one. The left most machine is going to be called message1 and the right most machine will be called message2. The back left one will be called monitor.
  + Message1:
    - IP: 169.254.86.44
  + Message2:
    - IP: 169.254.29.56
  + Monitor
    - IP: 169.254.159.18
* For the above IP’s to be correct both machines must be using local link
* To send the packets using the TCP protocol and encrypting with either AES 256 or SHA 256 we will be using a script that will allow packets to be sent to both machines.
  + The script can be found on the github link below
    - <https://github.com/Tyler-Wagner/Forward-Edge-Project>
* Tyler started a packet capture at 12:34 PM sending encrypted packets to both of the message machines.
* Possible attacks
  + DoS attack
    - Flood attack
    - Recourse exhaustion
  + Protocol-specific attacks
    - Take control of an existing SCTP association
    - Impersonate legitimate SCTP endpoints
  + Security Configuration
    - Weak authentication
    - Insecure parameter configuration
  + Packet Inspection and Tampering
    - Man in the middle
  + Day Zero exploits
  + Encryption Weaknesses
  + Vendor specific issues
* Small SCTP notes
  + Uses UDP aspects to transmit data, it does not break it down into blocks it only sends one big blob.
  + Secures the packets with a signature, kind of like how GPG keys work
  + Might be able to expose the signature but not the underlying encryption.
* Plan for 1/10/24
  + Start a DoS attack
    - Create a network and flood one of the computers with incoming packets causing resources to become strained, not allowing it to keep up with the amount of traffic coming in.
  + Read up on ARP poisoning and model how to do that attack while the DoS attack is happening. DoS will be the easiest to defend against since that is super common however ARP poisoning does not look like it is super defendable unless you already know it is happening.
  + Start reading the packet captures and figured out how to inject something into them. The code might have to get updated for this to start taking affect since right now all it is doing is sending packets and we will want to send files back and forth.

**1/10/24:**

* The far back left computer on the middle table got Ubuntu installed to start the DoS attack process.
* Set up the new netgear switch to mirror ports 1-8 on the switch to port 43
  + Default ip for the switch is 192.168.0.239 and password is the password
  + Tyler uses 192.168.0.240
* Hooked everything back up and started capturing packets on the two messaging machines and the monitoring machine.

**1/11/24:**

* Stopping the packet captures that were running all night. Nothing changed on the code yet just wanted more of a base line of what we are looking for.
  + Start adding packet captures to a separate folder on github so they can be stored somewhere.
* Going to upload the modified code which can be found at
  + <https://github.com/Tyler-Wagner/Forward-Edge-Project/tree/main/SCTP-Messages>
* I will pull the repo down onto the existing machines and start running the SCTP code to better mimic how the gen 1 boards were communicating
* We will need to finish finalizing the code to add encryption, sending files and updating the key every 3 (three) minutes.
  + Files will be sent in a round robin with the key being the last thing transmitted
  + The last 3 previous keys will be stored, we are trying to keep it the way it was described to us.
* While trying to update all of the necessary packages to get pysctp installed, I bricked message2 so I had to reinstall ubuntu during this process
* To go about sending files across the network and encrypting them I have updated the code and added a client-initiator and client-receiver.
  + Initiator: Generate 3 random 100 byte text files and send them in a round robin like fashion it takes 3 minutes to send the files.
  + Receiver: Its job is to get the 3 files then send them back to the starting point.
* The updated scripts can be found at
  + <https://github.com/Tyler-Wagner/Forward-Edge-Project/SCTP-Files>
* Plans for 1/12/24
  + Get the key exchange software working
  + Start attacking after the software is working

**1/19/24:**

* Below is an article to explain how the SCTP protocol works for those who are confused
  + <https://medium.com/@mytalk123/sctp-%E9%80%9A%E8%A8%8A%E5%8D%94%E5%AE%9A%E7%B0%A1%E4%BB%8B-6b30b74b3a73>
  + <https://en.wikipedia.org/wiki/Stream_Control_Transmission_Protocol>
  + https://www.ibm.com/docs/en/aix/7.1?topic=protocol-stream-control-transmission
* Starting to find any sort of vulnerabilities to start taking over SCTP since we know for a fact the boards will be using this to communicate with each other.
* The below article shows multiple ways to exploit the SCTP protocol
  + <https://www.rfc-editor.org/rfc/rfc5062>
* The below CVE allows the remote user to cause the target system to force a reload
  + <https://www.cvedetails.com/cve/CVE-2017-3826/>
  + If the target system does not have a limit on how much traffic it can receive then the above CVE will allow us to cause it to shut down. This is important since if we want to correctly simulate a proper ARP spoof where we declare a machine that is **NOT** the client or server and make it act as if it is, the target will need to be remotely forced to disconnect itself from the other system.
* The below article explains how we can use the SCTP protocol along with the ICMPv6 protocol to exploit an error message which will result in a kernel panic.
  + <https://www.freebsd.org/security/advisories/FreeBSD-SA-16:01.sctp.asc>
  + The reason a kernel panic would be a good way to shut the system down is because if the kernel shuts down then that means that the board itself will lose connection. While the key for initial transmission is stored if we can make it lose all previous keys stored the boards will more than likely not connect to each other any longer since the keys have fallen out of sync.
  + This could result in a possibility of a reply attack where we have an old key that we can then inject with whatever message we want
* The below link shows a CVE that we can use to continue to spoof IP addresses
  + <https://nvd.nist.gov/vuln/detail/CVE-2021-3772>
  + This is just another possible way to spoof either the client or the server to act as one or the other and connect.
* Below is a list of CVEs that Ubuntu 22.04 LTS has
  + <https://ubuntu.com/security/cves?q=&package=&priority=&version=jammy&status=>

**1/22/24:**

* Changed monitor’s network connection to work with DHCP so we can pull scripts. The new updated IP is:
  + 172.16.0.56
* Both monitor and monitor 2 are able to communicate now with the rest of the systems on the switch
* We are still looking for CVE’s based on Ubuntu 22.04 and SCTP as we know that is the operating system and protocol the gen 1 boards will be using.
* Starting to find scripts to minimize the amount of time it will take to perform attacks
* Installed parrotOS on both monitor systems to gain access to more tools
* Parrot bricked our systems, installed Fedora 39.
* Found CVE’s where we can start filtering through. All of it will be put in an Excel file and uploaded to the github. These notes will also be uploaded on github and taken off of here.
* Set up the two alienware machines to start programming on
* If the code can be emulated (not like android) then we can use that and find any sort of zero day issues with it.