Building an Isolated Wireless Lab Space on a College Campus

Final Report

Team 15
Client/Advisors: Doug Jacobson & Julie Rursch
Team Members/Roles:
Alec Sauerbrei — Curriculum Lead
Colin Ward — Communications Manager
Dalton Handel — Networking Lead
Hope Scheffert — Git/Documentation Manager
Omar Taylor — Software Design Lead
Tyler Much — Physical Design Lead
Team Email: sdmay18-15@iastate.edu
Team Website: http://sdmay18-15.sd.ece.iastate.edu/

Project Statement

Currently, wireless security courses here at Iowa State University utilize *wired* testing environments which generate “dummy” traffic so that students may observe and interact as if they are on a real network. This method is hard to scale, requires a great deal of equipment, and is also difficult to contain. Therefore, the proposed solution is for the creation of an isolated wireless facility for graduate and undergraduate coursework in wireless security.

In order to isolate these networks, a Faraday cage must be created which encapsulates the equipment such that only authorized users can connect to it. Additionally, these networks would need to be secured from the campus network using a proxy server so that undesirable traffic would not escape into the wild, and outside traffic will not be visible in the cage.

Another part of the project is to provide curriculum examples in the form of lab experiments which will be developed to highlight the educational value of these isolated networks.

Revised Project Design

The Faraday cage consists of two isolated networks. Firstly, the traditional 802.11 network is broadcasted through a router configured with a proxy which separates it from the Iowa State University network. The only devices on this network are two Raspberry Pi 3s that simulate traffic between one another, which can be observed through a WiFi dongle. Similarly, the cage encapsulates a Global Standard for Mobile communications (GSM) network. Connected to the router is a software defined radio (SDR) which acts as a cell tower and a sniffer. The GSM network traffic is automatically generated by two Android phones which call and text one another.

In order to access these environments, students must VPN to the Iowa State University network and access a virtual machine. The diagram below illustrates the design described above.

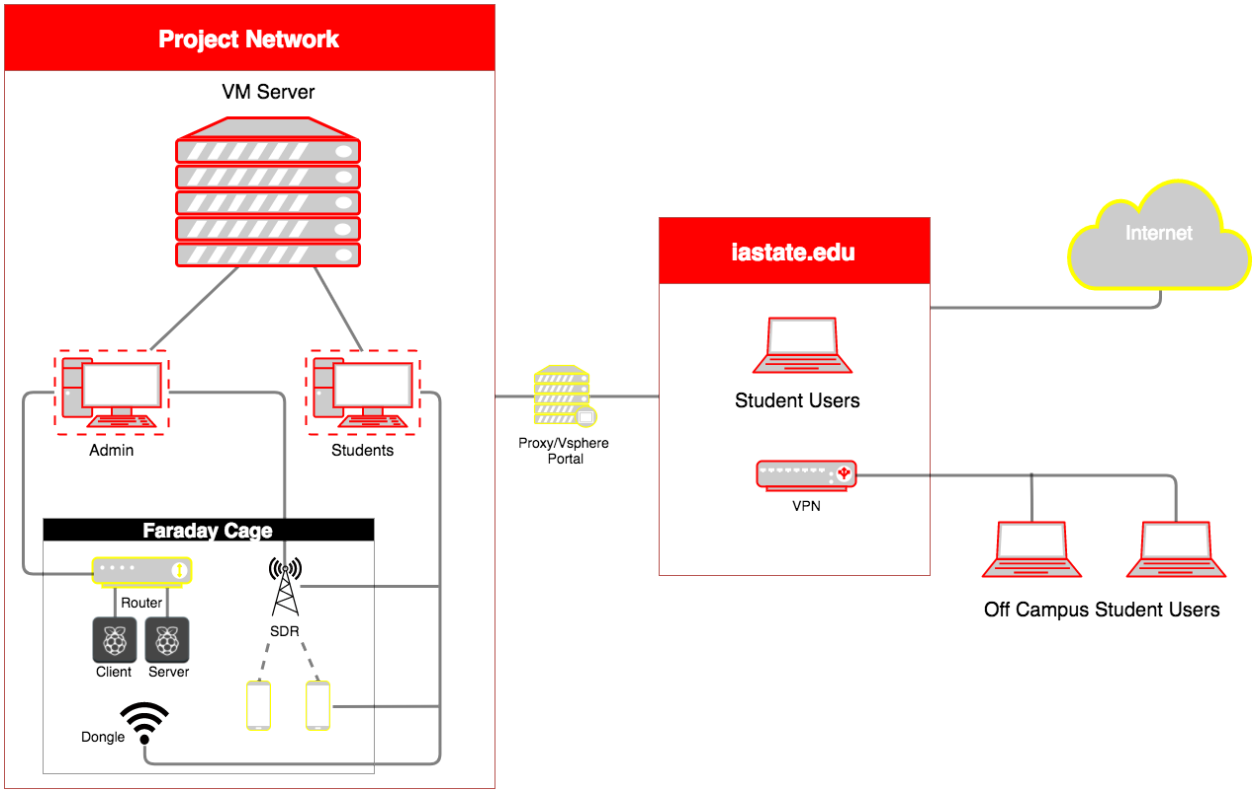

Figure 1

The Faraday cage is designed as a two foot by three foot plastic container with a removable lid. Inside, the container is completely lined with six layers of heavy duty aluminum foil. Then--to avoid the components touching the foil and boosting their signals--an additional container of the same dimensions was placed inside. The outside of the inner container was sprayed with MG Chemicals SUPER SHIELD Nickel Conductive Coating aerosol spray for further signal blocking strength.

A small “sub-cage” was assembled with 1mm steel mesh walls and copper foil tape on all edges. This was built specifically to surround the SDR and cell phones because the steel mesh material blocks GSM signals almost completely. The team also took two copper fabric sheets from the initial prototypes and placed them between the two containers where the aluminum foil is for added protection.

All power cords from inside the cage are routed through one drilled hole in the corner that feeds outside. All of the components are plugged into a power strip placed in between the two containers. Therefore only three cords have to be maneuvered through the hole: the WiFi dongle cord, the ethernet cable, and the power strip cord.

Below is a picture of the final design with all components mounted.

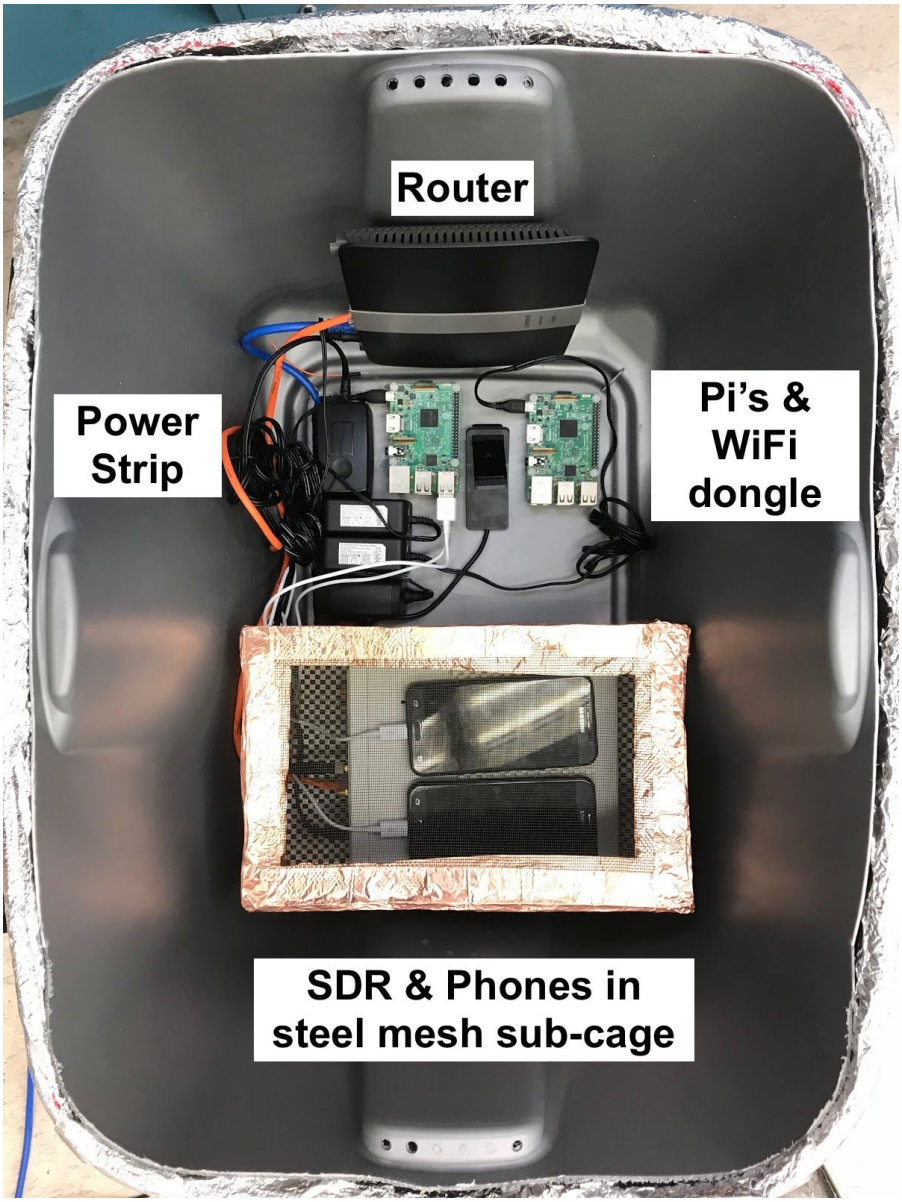

Figure 2

Implementation Details

Interface Specifications

The host server for the interior networks of the cage is directly connected via ethernet to the Iowa State University network. It is remotely available via VPN to the iastate.edu network, specifically at sd-may2018.ece.iastate.edu. The server orchestrates virtual machines (VMs) using VMware VSphere. Currently it contains three virtual machines for network administration: pfSense to implement NAT, OpenBTS Ubuntu for SDR configuration, and Admin for autonomous network activity. The rest of the VMs on the network will be for student activity.

Preemptively, around 50 virtual machines are needed to support the desired curriculum. However, the design emphasizes the ability to support more students if needed. These are to be managed by a designated administrator via the VSphere client.

From the host server, students will have access to a wireless dongle and the SDR. Depending on the lab requirements students may also have access to a Samsung J3. This allows them access to both a WiFi and GSM network. This access enables them to interact with any implemented curriculum activities.

Hardware

Though probably not considered “hardware”, arguably the most important piece of this project is the Faraday cage that blocks all other signals from entering or leaving the enclosed environment. This “cage” can be simply a shoebox wrapped with signal blocking fabric, such as the first prototype, or it can be a more durable solution like the final design previously described.

There are several hardware components mounted in the cage. The National Instruments USRP-2920 Software Defined Radio and Linksys AC1200+ Wireless Router are needed to broadcast their respective network signals within the cage. The Netgear N300 USB Adapter (WiFi dongle) is also needed as a sniffing point for students.

The cage contains two Raspberry Pi 3s and two Samsung Galaxy J3 unlocked Android phones. These components are vital to the testing phase because they are responsible for creating the traffic for these environments. The networks are useless to observe if no traffic is being generated within them.

Finally, the team obtained an ANTEC Server with 8 CPUs and 32 GB RAM which was then configured with VMware ESXi 6.5. This configuration allows the virtual machines to be set up on the server which in turn gives access to the components within the Faraday cage.

Software

The server runs VMware ESXi 6.5 for network management and configuration. On the server, there are several virtual machines. In order to allow internet connection to multiple VMs without having to register them all, there is a virtual machine that acts as an NAT router. It is called pfSense. It connects VMs connected to a virtual switch labeled “NAT” to the internet via the server’s ethernet connection to the Iowa State network. Next, there are a few VMs for wireless interaction within the confines of the cage. The first of them is an Ubuntu Linux 16.04 VM configured with OpenBTS. This VM is connected via ethernet through the router to the SDR. OpenBTS acts as the software implementation of a cell tower. It passes GSM traffic over UDP to the transceiver software which is sent to the SDR over ethernet and finally to corresponding GSM handsets (cell phones), all within the cage. Next there is a Kali Linux Admin VM. This has all necessary privileges and will be the only VM which can execute all network traffic scripts

discussed below. It is connected via a virtual switch labeled “UFC” to the router and the SDR (via the router) and has internet access via the pfSense VM. Finally, there is a Kali Linux Student VM which will be used by students in future wireless security courses at Iowa State to interact with and observe the wireless networks in the cage created by the router and SDR. This VM is to be cloned to support as many as 50 students at this time.

On the Admin VM, there are a set of Python and bash scripts to autonomously generate network traffic so that students have packets to sniff. Raspberry Pi scripts include logging in and out of Facebook and sending emails using the Pi’s email server. Android phone scripts include calling and texting the other phone to simulate a real cellular network.

Implementation Issues and Challenges

Signal Blocking

As stated many times, the challenge of this project was creating a cage that will completely trap its own signals in while blocking all others. The first prototype was not as successful as the team had hoped, but the second was more promising. Nonetheless, precaution was needed when designing the final product to ensure two secured networks.

It took some time to finally figure out which materials blocked which signals the best. It turns out the 1mm hole steel mesh worked very well for blocking cellular signals, but had little-to-no effect on the WiFi signals. The best option for blocking WiFi signals was the combination of the aluminum foil and the Nickel Conductive Coating.

Android

Another interesting challenge came from creating scripts for Android phones to automate phone calls and SMS text messages. This topic is not very common, so it required some research and experimentation. The first idea was to create a custom Android app which sent SMS messages and made phone calls based on a timer. However, this was not ideal because the app would always need to be running on the phones. After more research, Android Debug Bridge (ADB) was found to be a useful tool for this task. This allows the use of simple shell commands over USB to the Android phone which is placed in Debug mode. Integrating these in a simple bash script creates the perfect simulation of a conversation.

Additionally, Android does not allow the phone to play any audio files during a phone call, as this is a security risk. Instead, in attempt to simulate a realistic phone call, the team decided to use Google’s Text to Speech utility. This is called directly from the bash script and is played out of the Raspberry Pis. Although it is not ideal to have a speaker playing audio files aloud in the cage, it does not affect anything in the environment and it gets the job done.

Cellular Network

There were a number of challenges experienced when setting up the cellular network component of this isolated wireless lab space. Iowa State has not historically taught any courses on cellular networks, and a small amount of information about such networks has been made publicly available. This made the task of gathering information about building a cellular network a challenge in and of itself. The first of these challenges dealt with finding hardware that could act as the enclosed network's cellular base station. Purchasing a software-defined radio (SDR) proved to be an expensive solution so a SDR already owned by Iowa State was used as an alternative, a NI-USRP 2920.

The next task was to find an open source GSM software implementation to work in coordination with the radio. Deciding on OpenBTS as GSM air interface was less of a challenge compared to the realization that the software did not list the chosen SDR as a supported radio. Fortunately, a third-party guide was found that walked through the deviations needed to be made from the official guide in order to properly build OpenBTS and connect it to the SDR.

After many hours spent building OpenBTS onto an Ubuntu Server 16.04 virtual machine, it was time to establish a connection from the OpenBTS host machine to the SDR. At this point in time, the host machine was a personal machine instead of the ESXi 6.5 server. With such little hand-holding available, establishing this connection proved to be one of the most challenging portions of the project. Normally, after installing the required software and drivers for the SDR, the ethernet adapter should have been configurable to be on the default network of the SDR such that we could ping the radio, but there was no response. After many hours researching this issue, it was found that there was a way to reset the device to factory settings by putting the device in a Safe Mode. This yielded fruitful results as a connection with the radio was able to be established, although only for temporary period of time. As soon as the radio was taken out of Safe Mode, it would boot back into the same state that disabled itself from being queried. It was at this point that the true complexity of the situation had presented itself and services of someone with more experience working with SDRs was requested.

The graduate student consultant, Paul Pfister, had worked with the SDRs being used for the project before and guessed the IP addresses may have been configured to operate on the Iowa State network. After recommending utilizing Wireshark to sniff packets coming from devices on that subnet, the SDR was found and could finally be reconfigured and updated outside of Safe Mode, allowing the changes to be permanent.

It was time to move the OpenBTS virtual machine onto the ESXi server and configure the network group OpenBTS operated under to be able to establish a connection to the SDR through the router placed inside the Faraday cage. After many failed attempts to establish a connection, the problem was solved by properly configuring the router and right OpenBTS VM

network interface to be the same subnet as each other. Alas, OpenBTS could establish a connection from the server machine to the SDR via the router.

The final issue encountered was related to interacting with the now established cellular network. Phone call tests were needed to make sure it worked. However, to do this, an unlocked phone with a vender unlocked SIM card was required. The unlocked phone had already been acquired during the early stages of the project, however acquiring such a SIM card proved to be an expensive and logistically infeasible ordeal. To deal with this, a U.S. cellular SIM card one of the project members had on hand was used. It was cut to fit into the phone and an attempt to connect to the OpenBTS network was made. While the phone was able to see the network, the connection attempt was unsuccessful. Further progress on this front has not been made, but there were a number of follow-up steps that could be taken solve this issue.

Server

Due to a scarcity of experience with configuration of server machines amongst the team, setting up the host for the VMs was another hurdle. The first issue the team faced was interfacing the internal network with the external adapters. In order to manage this, a series of virtual switches were configured. These allow for groups of VMs to all have a connection to one of the external adapters and to other VMs. There were two external adapters of concern. The first of which was the ethernet connection to the Iowa State network. The only VMs connected to this switch are the pfSense VM for distributing internet access and the admin kernel which sets up the VSphere portal for remote connection. Both of these VMs needed their MAC addresses registered with the Iowa State network in order to properly operate. The second adapter is set up for access to devices within the cage. The main challenge with these adapters was determining the necessity to have them on separate virtual switches, as well as the realization of what the virtual switches were used for on the server.

The need for internet access to be common amongst cage devices and VMs also proved to be difficult. It is important to have internet access for the VMs for maintenance, and the cage devices requires internet access to generate network traffic. It is impractical to have to register each individual VM, thus there was a need to share an IP address among the VMs placed on the server. To accomplish this, a NAT router VM using pfSense was built, configured, and registered with the Iowa State network. It is connected directly to the Iowa State network via a virtual switch and also to a virtual switch with no outward interface. This virtual switch (labeled “NAT”) connects other VMs to this router. Thus, each VM is placed in this subnetwork to borrow an IP address from the pfSense VM and communicate through the pfSense VM’s connection to the Iowa State network to reach the internet.

Testing

Functional Testing

With respect to cage testing, the procedure is simple. The wireless and cellular components are placed inside and the cage is sealed by placing the lid on top. The team then attempts to connect to the networks inside to see if the signals are being sufficiently blocked.

Further, the cage needs to successfully trap its own signals. This means that the WiFi network broadcasted within the cage should not be accessible or even visible from outside the cage. To test this, the team stands next to the cage and tries to see if the SSID of the cage's router is visible on their devices. This test especially fails if they are able to connect to the network inside the cage.

Similarly, the cage cannot allow any bystander to connect to its cellular network. In fact, failure to isolate the SDR's signals would mean breaking the law, as it is a spoofed cell tower. Fortunately, testing shows a complete loss of cell signal, with tests coming back as -120 dB signal strength consistently for cell signal--therefore, this is not any concern.

Android phones were connected to the SDR's GSM network. The scripts automate calling and sending SMS text messages were first tested outside of the cage with team member phones to ensure expected behavior.

To get actual metrics, the team found a "Faraday Cage" mobile app that can be run on a phone to detect changes in signal strength for both WiFi and cellular networks. This app was used by first starting the test, placing the phone inside the cage, sealing the cage, and waiting. After roughly 30 seconds, the cage was opened and the phone was taken out to review the results of the test. The results include a graph of the signal strength measured in decibels (dB) and time (sec), as well as total signal blockage. To get an idea of the meaning of the decibel scale in terms of signal strength, a graph is provided below. The team goal was to reach at least -110 dB on this scale.

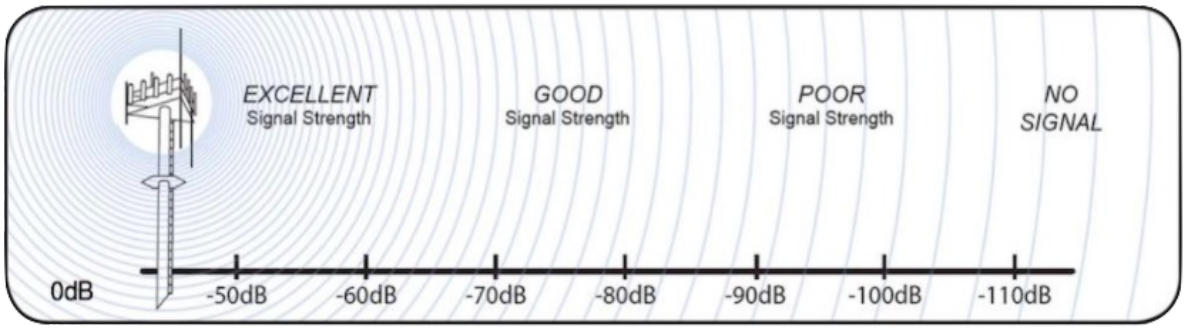

Figure 3

Source: "How to Measure Signal Strength in Decibels on Your Cell Phone?" *SignalBooster.com*, 4 Aug. 2016, www.signalbooster.com/blogs/news/how-to-measure-signal-strength-in-decibels-on-your-cell-phone

Below are the final results taken straight from the Faraday Cage app. WiFi signal is shown in blue and GSM signal is shown in yellow.

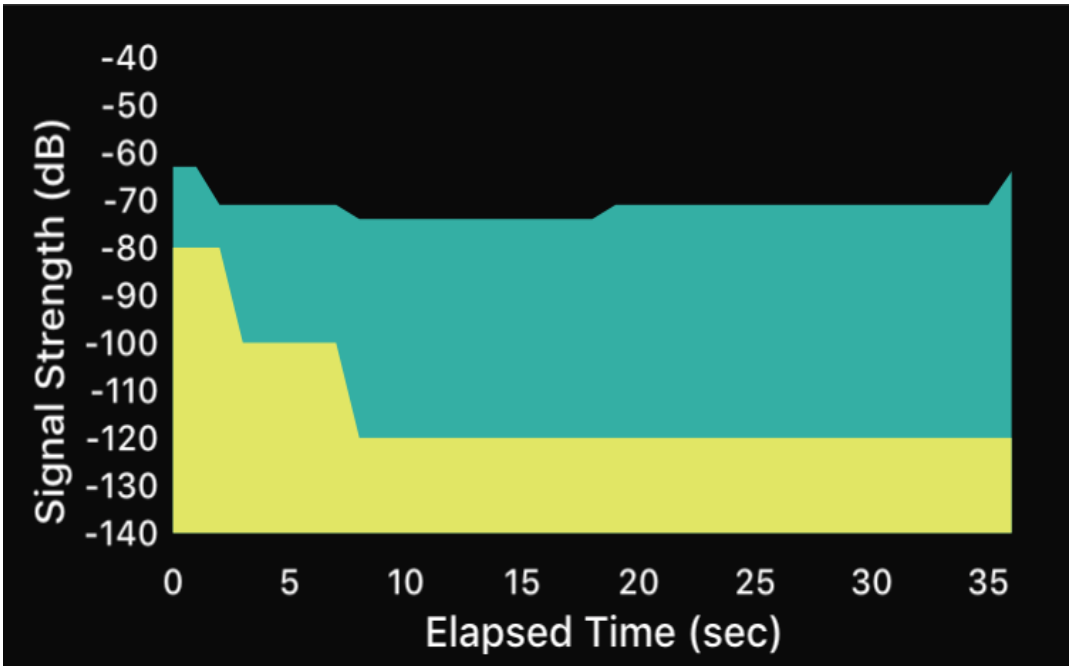

Figure 4

The graph above shows the effectiveness of only the steel mesh sub-cage surrounding a team member's phone. After about 10 seconds, the cellular signal (in yellow) is blocked according to Figure 3.

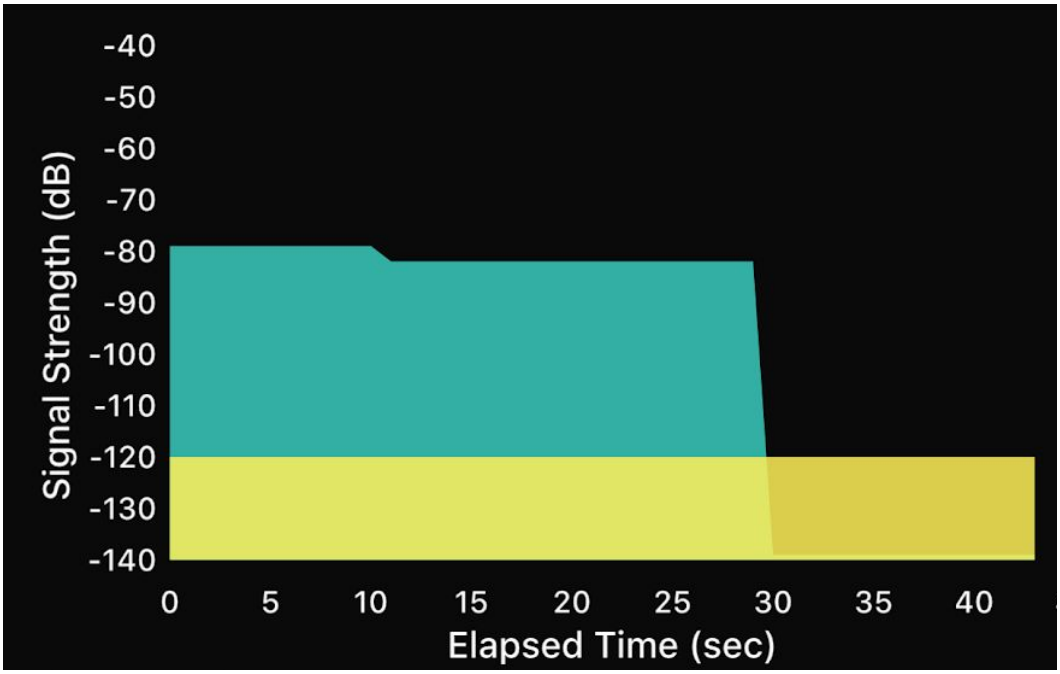

Figure 5

Even more impressive, when the cage is sealed, the WiFi signal (in blue) is completely dropped after 30 seconds.

Based on these results, it is safe to say that all signals are sufficiently blocked and the team goal has been achieved.

Final Testing Results

Test	**Passed (✔/X)**
Failure to contact devices inside the cage	✔
WiFi network does not allow any devices outside of the cage to connect	✔
GSM network does not allow any cell phones outside of the cage to connect	✔
Achieving at least -110 dB in signal strength measurement (no effective signal)	✔

In order to test that the networks are generating sufficient traffic, the team accessed the environments through virtual machines and used Wireshark to sniff packets, just as students will be doing in future labs. Once operating in a lab setting, there will be scripts that constantly run

to generate packets at a reasonable rate so that at any point when a student goes to work, there will be an active environment in the cage.

Non-Functional Testing

Students must log in to the Iowa State University network via VPN in order to access the virtual machines, therefore security is not a main concern. VPN access was tested using team member Iowa State University credentials. Because these tests were successful, it was sufficient for other student access. Access was tested on and off campus, as this should cover all possible connection use cases that students will experience.

If students are trying to learn through observation of these networks, it is important that they produce enough traffic in a sufficient time frame. It is a team goal that there will be network traffic to observe at all times. This way, students avoid having to wait for the chance to sniff packets and perform lab exercises.

Related Projects

Individuals from Dakota State University have been developing a product similar to the cellular Faraday cage proposed here. They have shared ideas regarding what they have used in their Faraday cage such as its cellular base station, what phones have worked in the cage for simulating traffic, and some hurdles that still need to be overcame. Below lists information received by the advisor of the project in early September.

- Cellular base station: They use OpenBTS with an Ettus B200 radio. It works well, but there are many challenges to getting the software set up. They were happy to help and offered to provide a preconfigured VM, however the team decided to do this internally.
- Phones: Some phones are better than others for actually connecting. Many students with Android phones can easily connect onto the new cellular network without any configuration or hardware changes. Those with iPhones will not be able to connect without replacing the SIM card in their phones with ones that are pre-configured. Most 2G/GSM style phones should function.
- Isolation: In their in-classroom setups, they generally remove the antennas from the radios, which limits the range. The problem is that the cell phones have a very high powered radio that will reach a great distance. For their final version of the lab (where it will be accessible to others from the internet), they are working on Faraday cages. There is also the issue of heat dissipation. They are ordering a custom server rack built by HM Cragg that will be able to handle the servers and store the equipment while allowing ventilation.

This information was helpful because it allowed the team to narrow down research for compatibility of the SDR and to use strictly Android phones. It was also an appreciated warning about the signal strength of the SDR.

It is possible to purchase premade Faraday cages, and this was considered at the beginning of the project. Faraday bags that fit individual phones range in price from $50-$80, with setups that could enclose the entirety of the project components ranging from $300-$1,000. However, the team liked the challenge of creating a custom cage, and saving the client-advisers money was also a plus.

Conclusion

Over the last two semesters, the team designed, prototyped, built, and tested a Faraday cage which isolates two different types of wireless networks. The team is confident that the cage does its job and that the networks will serve as excellent learning environments for future wireless security courses at Iowa State. In fact, because this cage is so easy and inexpensive to build, other universities may be interested in replicating this solution.

With the construction of this cage and automated scripts to generate traffic, students will be able to observe and learn about wireless and cellular network security without worrying about breaking the law. The ability to manipulate the cellular network alone is truly valuable because this is not legal in the real world.

It is exciting that this small proof of concept gives professors the option to build upon their curriculum to include GSM network and wireless security, and even possibly experiment with the interaction between WiFi and GSM networks.

Appendix I: Operation Manual

Network Access

Below are instructions on how to access the cage networks.

1. Connect to the Iowa State University network (this can be done via VPN)
2. Visit https://sd-may2018.ece.iastate.edu/ui/#/login
3. Log in to VMware ESXi with necessary credentials
4. Go to Virtual Machines
5. Choose a VM
6. Log in to VM with necessary credentials

Traffic Generation

Below are instructions on how to start automated traffic generation within the networks.

1. See Network Access instructions above to access the Admin VM
2. Once logged in to the VM, enter the Traffic_Scripts directory
3. To automate the process of logging in and out of Facebook:
   `./run_fb_login_logouts.sh`
4. To automate the process of logging in and out of an arbitrary website (this must be edited according to that website name and HTML attributes):
   `./run_website_login_logouts.sh`
5. To automate sending emails over R2DPi's email server:
   `./run_send_smtp_emails.sh`
6. To automate SMS texts between the two phones:
   `./run_automated_sms.sh`
7. To automate phone calls between the two phones:
   On phone one: `./run_outgoing_calls.sh`
   One phone two: `./run_incoming_calls.sh`

OpenBTS

Below are instructions for setting up the SDR using OpenBTS.

1. Start all the required services for OpenBTS on the Ubuntu Server:
   ```
   sudo start sipauthserve
   sudo start smqueue
   sudo start openbts
   sudo start asterisk -vvvv
   ```
2. Run osmo to allow OpenBTS to communicate through the SDR transceiver
   ```
   osmo-trx -f
   ```

3. Run OpenBTS
   ```
   cd /OpenBTS
   sudo ./OpenBTS
   ```
4. Check network setup by using Open Registration and making a test call.
   a. In OpenBTS console:
      ```
      config Control.LUR.OpenRegistration ".*"
      config Control.LUR.SendTMSIs "1"
      rxgain 20
      ```
   b. On phone go to Settings->Mobile Networks->Network Operators and choose 00101
   c. Call 2600. It is the OpenBTS echo service (think of this as a ping to the SDR from the phone).

Appendix II: Initial Designs

Initial Prototype

After receiving a sample of the copper Faraday fabric, it was wrapped and secured around a shoebox to create an extremely simple initial prototype. Simple tests were conducted with a cell phone in the box and it was determined that, unfortunately, the fabric alone was not enough to adequately block all signals. However, the team considered that another layer of the fabric may increase the blocking power and lead to a better solution. Below is a diagram of the initial prototype and tests.

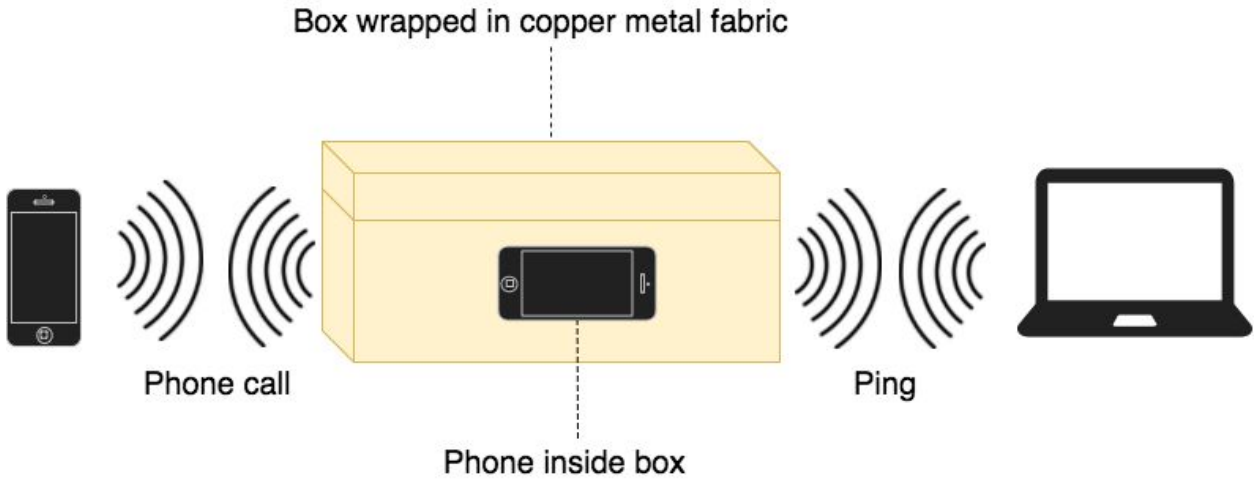

Figure 6

Reinforced Prototype

Due to the lack of signal blocking of the initial prototype, the team reinforced it with additional signal blocking materials. Firstly, pieces of 1mm steel mesh were stapled to the inner walls of the box. This only slightly improved the blocking power. After speaking with Dr. Mani Mina, the team wrapped the box with one layer of heavy duty aluminum foil, as he suggested. This significantly improved the signal blocking, leading the team to believe this was the design to be used for the final cage. Below is a diagram of the reinforced prototype and tests.

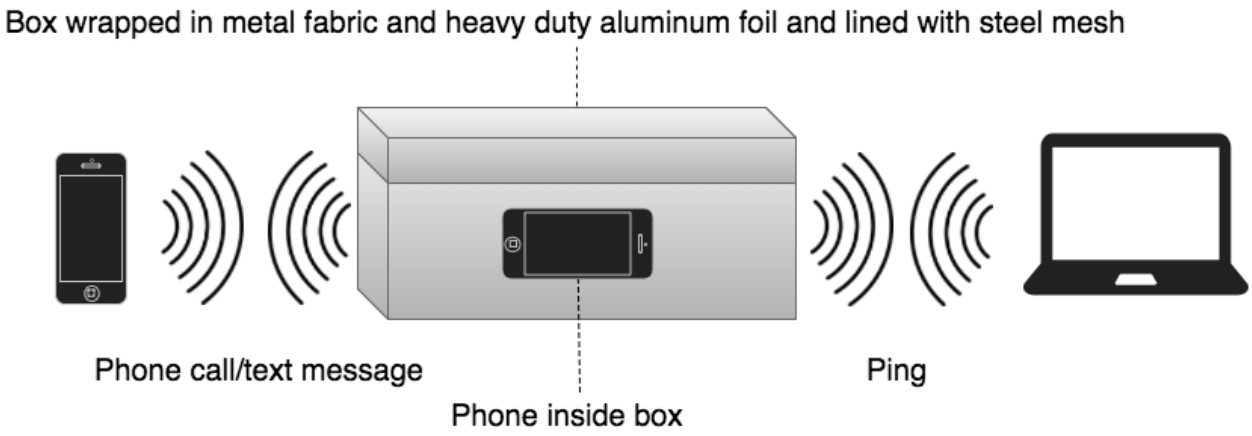

Figure 7

Appendix III: Other Considerations

Faraday Cage Materials

The following materials were needed to build the final Faraday cage:

1. Two 2ft x 3ft plastic storage containers with lids
2. Three rolls of Heavy Duty Aluminum Foil
3. One 3ft by 3ft steel mesh sheet with approximately 1 millimeter sized holes
4. One can of MG Chemicals SUPER SHIELD Nickel Conductive Coating aerosol spray
5. One roll of Copper Foil Tape Conductive Adhesive

Project Theme

Throughout the project, various hardware and accounts needed to be named. The team decided to name devices and accounts based off of Star Wars characters. Some of the names are obscure references, but the correlation was drawn based on what the character did in the movies and what the function of the object was. For example, the router is called DroidControlShip and the two Raspberry Pis are named R2DPi and C3PiO. The team thinks this will fit well with the other themed systems the client-advisors currently use (Star Trek).

Appendix IV: Code

The following github link contains the team code repository. Inside you will find Java code that has been developed for the SMS application (although this idea was scrapped) and code relevant to curriculum development. You will also find Python and bash scripts to control GSM communications and simulate network traffic.

GitHub: https://git.ece.iastate.edu/sd/sdmay18-15