# Roll Call Software

## Sponsor: Federal Communications Commission | Team: 3B&T

## INTRODUCTION

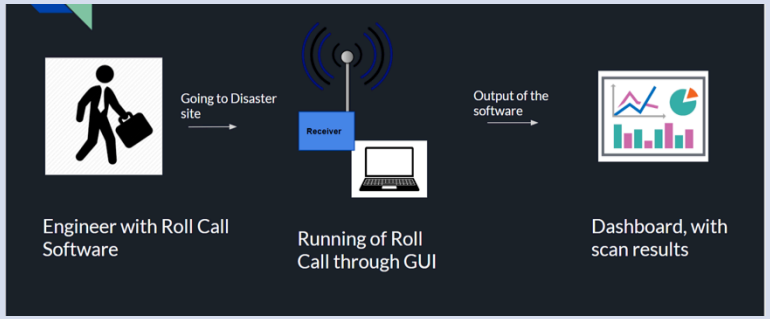
- The FCC uses a system called Roll Call to provide data about communications systems that may have been impacted by natural disasters.
- Impacted communications systems include AM/FM radio stations, cell phone towers, and emergency broadcast systems.
- The goal is to provide other agencies, such as FEMA, with the necessary information to begin restoring these communications systems.
- Challenges: Roll Call has historically operated in a command line interface (CLI), which is not user-friendly. Additionally, all post scan results were placed in a zipped folder, intended to be sent off immediately

## OBJECTIVES AND REQUIREMENTS

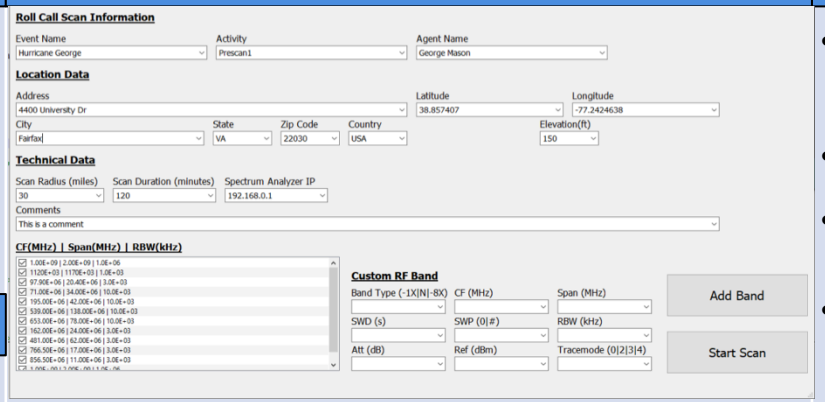
- The CLI has been replaced with a Graphical User Interface (GUI) in the Windows environment to provide a more user-friendly format.
- A dashboard was developed to provide post scan data results within one contained window on-site.
- We included a database, attached to the Roll Call software, that holds data from previous scans and supplies the GUI with the 15 most recent scans
- These functions do not rely on an internet connection

## CONCEPT OF OPERATIONS

Before and after a disaster, an engineer travels to a disaster site with the Roll Call software and equipment. The engineer then runs the Roll Call scan through the GUI. After the scan, the dashboard allows the user to view the scan results as needed.



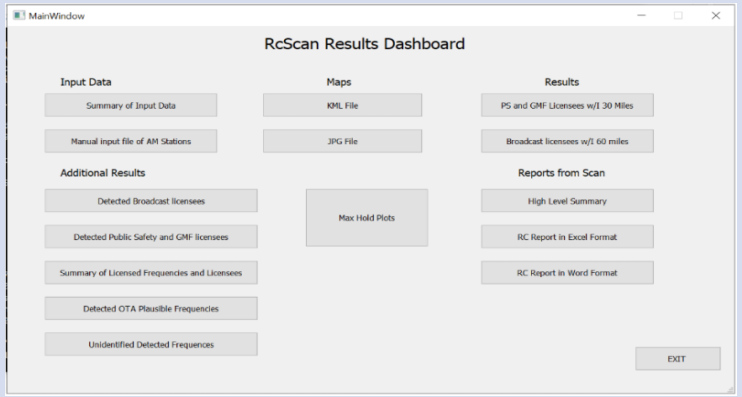

## DESIGN




The GUI shown above takes in pre-scan information inputted by the user through Pythons cross-platform GUI toolkit Qt (PyQt5)and allows the engineer to verify all input information before beginning the scan.

At the end of each scan, the database (SQLite3) is updated and scan results are presented in a dashboard on-site. The database holds all user input utilized in scans and allows engineers to access 15 previous scan's inputs to simplify repeated scans. The GUI takes user input parameters that are then utilized for the scan.

When the scan is complete, a dashboard containing the most relevant data is displayed on-site in a pop-up window depicted in the figure below, while an XML file containing detailed post-scan data is transmitted to FCC headquarters.




## VERIFICATION AND VALIDATION

- The software was placed on another laptop in order to implement changes without impeding the previous software on the laptop from the FCC. The changes were then made until all issues encountered during testing were resolved. The new version of the software was then replaced with the previous version of the software on the FCC laptop.
- Functional testing was completed on the GUI, dashboard, and database to ensure these components would operate as intended, even in the presence erroneous user input.
- Database testing was composed of long term data simulations; this was completed through the creation of a Python script that would fill the database with roughly 30,000 entries. This testing was also utilized for performance checks in the database query.
- For the database, SQL Injection attacks were considered. We implemented sanitizations techniques to prevent user input fields from accepting SQL characters that are considered hazardous. Only SQL safe characters are accepted from user input to prevent SQL injection attempts from leaving the GUI, ultimately protecting the database.

## CONCLUSIONS

In conclusion, radio frequency (RF) communications are an important part of disaster relief and need to be available after a disaster occurs. Without access to these communications systems, including AM/FM radio, cell towers, and emergency broadcast systems, individuals and responsible agencies can no longer obtain potentially critical information in a quick and effective manner. Therefore, the ability to scan for these communications and figure out where they went down is an important process that can be life saving.

Ultimately, these communications represent a crucial safety priority during disasters, and ensuring these methods of communication remain active after a tragedy.

## ACKNOWLEDGEMENTS


The members of 3B&T would like to thank the FCC for sponsoring this project. Every week, they took time out of their schedules to meet with us so that we could communicate our progress and they could provide us with feedback.

We would also like to thank our mentor, Dr. Powell, for guiding us through this entire process. He did his best to make sure that our project was done on time and the best it could be. The experience we gathered from this project is invaluable and will undoubtedly help us in our future careers.

# No Equipment Failed/No Malicious Intent (NEF/NMI) Cyber-System Failures

Alyssa Lopez, Sam Faulconer, Hissah Almarzook, and Edwin Padilla-Gonzalez
Cyber Security Engineering Department, George Mason Univeristy

Systems Engineering and Operations Research

GEORGE MASON UNIVERSITY

## Introduction

No Equipment Failed/No Malicious Intent (NEF/NMI) cyber-system failures are failures caused by unforeseen complications that arise from changes made to complex systems. NEF/NMI failures occur with no equipment failure nor a malicious actor. Organizations that need to apply updates or make configuration changes to their system(s) will inevitably experience a NEF/NMI cyber-system failure if the proper precautions are not taken to prevent them. The results of these failures can range anywhere from inconvenient to catastrophic.

This research was conducted based on a case study of a NEF/NMI failure at the Hatch Nuclear Power Plant (NPP). In this example, a nuclear engineer connected his work computer to a reactor control system for remote monitoring and administration. This configuration change allowed the remote monitoring and administration software to push an unauthorized software update to the reactor control system. This update caused the reactor control system to reboot and clear its historical data of reactor coolant levels. Once other safety systems observed the very sudden drop in coolant levels, a full shutdown of one of the plant's two reactors was initiated.

## Concept of Operations

In order to test and prevent NEF/NMI complex system failures a Systems Integration Engineer (SIE) will require the probability that a failure will occur, as well as the impact this failure would incur. The SIE will be responsible for the ensuring components/ software are introduced to the overall system safely. In order to determine the probability of system failure in a NPP, the SIE will query the Common Critical Failure (CCF) database for failures involving the system that is to be modified. The CCF database collects failure data in commercial NPPs so a SIE can search for failures involving the system/component being modified. The SIE will take the results of this query, as well as their expert knowledge of the system to determine a probability the modification will cause a failure. The SIE will then determine the systems criticality based on its classification as one of the following.

- High: digital assets associated with plant safety and trip functions/communication.
- Medium: digital assets that do not functional primarily as safety systems, but may affect plant safety in their operation.
- Low: independent assets that do not affect plant safety or trip functions/ communication.

The probability and criticality of the system/component being modified are illustrated in the risk matrix in Figure 1 and are used as input values for the Decision Support System.

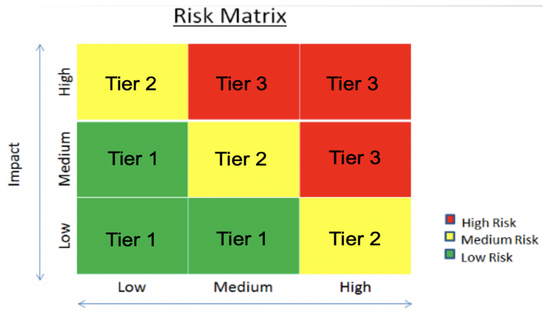



**Figure 1**

## Design

The DSS was built as an Activity Diagram using Innoslate. The Activity Diagram format shows the overall flow of control of the DSS. The Activity Diagram format was able to clearly depict the different decisions, shown as diamonds in Figure 2, that a user needed to make regarding the likelihood and impact of a proposed modification to a BWR system. The Activity Diagram also clearly shows the actions taken by the user, shown as rounded rectangles in Figure 2.

The DSS is based on three different options for each the likelihood of failure and the impact of failure: low, moderate, and high. Since the decisions in the Activity Format are binary, the design of the DSS had to be slightly modified. For example, the first decision node prompts the user to decide if the proposed modification to the system has a low likelihood of failure. The user is able to select "low likelihood of failure" or "not low likelihood of failure." The "not low likelihood of failure" option would lead the user to be prompted to decide if the proposed modification to the system has a moderate likelihood of failure. Here, [W4] the user is able to select either "moderate likelihood of failure" or "high likelihood of failure." The impacts of potential failures were addressed in the same way to fit the binary representation of decisions in the DSS.

The output of the DSS will be the required testing that a system modification must go through before it may be safely implemented. The probability of failure and the system in question criticality directly influence the standard of testing that must be implemented. At the highest level, Tier 3, statistical testing must first be done to provide sample data to a subsequent white box test. This will use the provided sample data to execute each branch of the software in the system/component being modified. Next, Tier 2 testing can be done. This includes using sample data, either from previously conducted statistical test or system/component specification, to conduct a black box test. Finally, Tier 1 specifies that the system/component pass a functional test to ensure that the system/component operates as intended without failing.

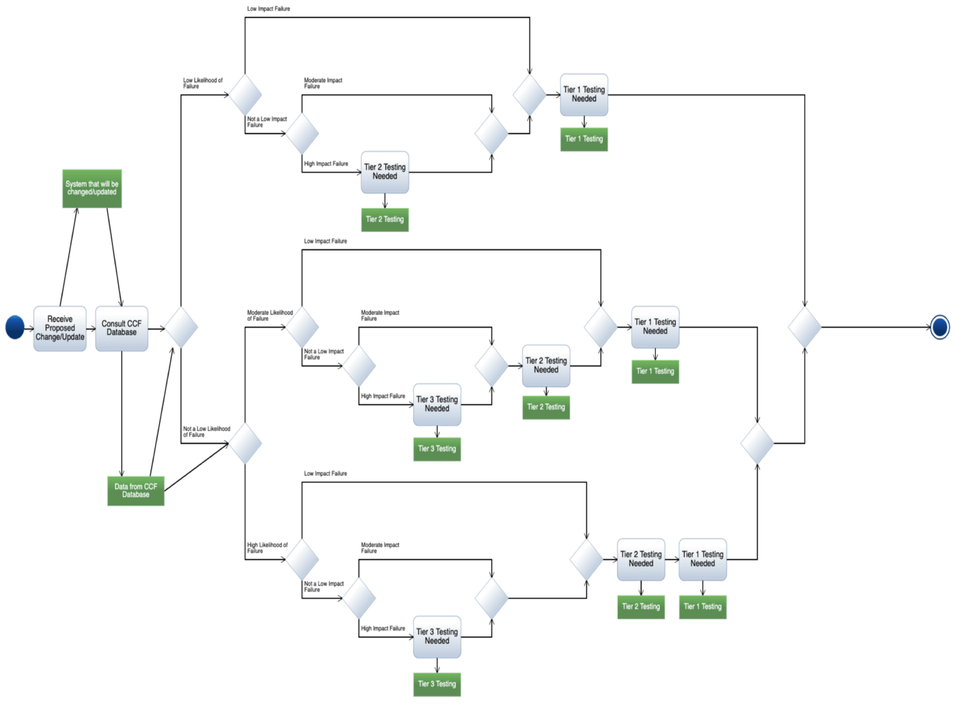



**Figure 2**

## Verification

The tiers of testing that the DSS outputs are based on the levels of risk associated with proposed modifications to the BWR system. Because the information contained in the CCF database is proprietary information, the DSS was unable to be tested with actual probabilistic inputs. However, if the correct inputs are used in the DSS for the likelihood and impact of failure based on the CCF database query, the DSS will output the correct testing tier that is needed for the proposed modification.

While the DSS was unable to be tested, the concept of testing proposed modifications based on the level of risk that they pose a risk on the BWR system could help lower the likelihood of NEF/NMI failures in BWR systems. Performing the testing that the DSS outputs will allow engineers to understand how BWR system components react to the proposed modification. By understanding how BWR system components react to proposed modifications prior to their implementation in the operational environment, the engineers at the plant will have a better understanding of how the proposed modification will affect the BWR system. The results of testing the proposed modification to the BWR system will determine whether the Systems Integration Engineer will implement the proposed modification in the operational environment

## Conclusion

NEF/NMI cyber-system failures can occur if the proper precautions are not taken when implementing modifications to a system. The Edwin I. Hatch nuclear power plant experienced a NEF/NMI failure due to a configuration change that did not go through prior testing to indicate that the configuration change was safe to implement. The solution provided above is robust enough to prevent this failure and similar BWR NEF/ NMI failures. If the engineer at the Edwin I. Hatch NPP had tested the configuration change prior to implementing it in the operational environment, the engineer would have been able to see how the BWR system would have interacted with the configuration change.

The DSS was designed to aid organizations in making risk-based decisions regarding proposed modifications to a BWR system. This is done by utilizing data-based probabilities and standardized risk categories to determine different tiers of integration testing. The DSS accepts a proposed modification to the system and data from the CCF database as an input. The outputs are the tiers of testing that should be conducted based on established industry standards. The results of the testing can aid plant engineers in determining if it is safe to proceed with the given modification.

The DSS that was created is specialized to be used in a BWR NPP, but further research can be done to apply the concept behind the DSS to other complex systems to help prevent NEF/NMI failures in other systems.

## References

1. [1] B. I. Spinard and W. Marcum, "Nuclear reactor," *Encyclopædia Britannica, inc.*, Sept. 2019. Accessed on: Mar. 26, 2020. [Online]. Available: https://www.britannica.com/technology/nuclear-reactor
2. [2] M. Brain, R. Lamb, and P. J. Kiger, "How nuclear power works," *HowStuffWorks.*, Oct. 2000. Accessed on: Jan. 2020. [Online]. Available: https://science.howstuffworks.com/nuclear-power.htm
3. [3] "License Amendment Request Guidelines," *Nuclear Energy Institute.*, Oct. 2010. Accessed on: Feb. 2020. [Online]. Available: https://www.nrc.gov/docs/ML1033/ML103360404.pdf
4. [4] "Cyber security programs for nuclear facilities," *U.S. Nuclear Regulatory Commission.*, Jan. 2010. Accessed on: Jan. 2020. [Online]. Available: https://scp.nrc.gov/slo/regguide571.pdf
5. [5] M. Caruso, M.C. Cheok, M. Cunningham, G.M. Holahan, T. King, G. Parry, A.M. Ramey-Smith, M. Rubin, and A. Thadani, "An approach for using probabilistic risk assessment in risk-informed decisions on plant-specific changes to the licensing basis," *Reliability Engineering & System Safety.*, vol. 63, no. 3, pp 231-242, Mar. 1999. Accessed on: Jan. 2020. [Online]. Available: doi: 10.1016/S0951-8320(98)00038-6
6. [6] B. Krebs, "Cyber incident blamed for nuclear power plant shutdown," *The Washington Post Company.*, Jun. 2008. Accessed on: Nov. 2019. [Online]. Available: https://www.washingtonpost.com/wp-dyn/content/article/2008/06/05/AR2008060501958.html
7. [7] "Federal, state, and local, and tribal, responsibilities," *U.S. Nuclear Regulatory Commission.*, Jun. 2018. Accessed on: Jan. 2020. [Online]. Available: https://www.nrc.gov/about-nrc/emerg-preparedness/about-emerg-preparedness/federal-state-local.html
8. [8] T.E. Wierman, D.M. Rasmuson, A. Mosleh, "Common-cause failure database and analysis system: event data collection, classification, and coding," *U.S. Nuclear Regulatory Commission.*, Sept. 2007. Accessed on: Feb. 2020. [Online]. Available: https://www.nrc.gov/docs/ML0729/ML072970404.pdf
9. [9] J.W. Lee, C.K. Lee, J.G. Song, K.C. Kwon, and D.Y. Lee, "A cyber security risk assessment for the design of I&C systems in nuclear power plants," *Korea Atomic Energy Research Institute.*, vol. 44, no. 8, pp. 919-928, Dec. 2012. Accessed on: Jan. 2020. [Online]. Available: doi: 10.5516/NET.04.2011.065
10. [10] G.S. Bedi, "Guidelines for inservice testing at nuclear power plants," *U.S. Nuclear Regulatory Commission.*, Oct. 2013. Accessed on: Jan. 2020. [Online]. Available: https://www.nrc.gov/docs/ML1329/ML13295A020.pdf
11. [11] V. Grover, W.J. Kettinger, *Process Think: Winning Perspectives for Business Change in the Information Age.* " Hershey, PA: IGI Global, 2000.
12. [12] "Systems Integration," *The MITRE Corporation.*, Mar. 2016. Accessed on : Apr. 2020. [Online]. Available: https://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/systems-integration
13. [13] *IEEE Standard for Software and System Test Documentation*, IEEE Standard 829, 2008.
14. [14] "BWR14 technology manual (R-104B)," *USNRC Technical Training Center.*, Accessed on: Mar. 2020. [Online]. Available: https://www.nrc.gov/docs/ML0228/ML022830867.pdf
15. [15] "Risk-informed categorization and treatment of structures, systems, and components for nuclear power reactors," *U.S. Nuclear Regulatory Commission.*, Aug. 2017. Accessed on: Mar. 2020. [Online]. Available: https://www.nrc.gov/reading-rm/doc-collections/cfr/part050/part050-0069.html

# Phishing Susceptibility and Analysis Study

Te Ming Tiong, Kristin DiMichele, Abdulla Alhamer, Abdullah Alsaadi
Professor Gino Manzo, Dr. Kathryn Laskey, James Lee
Cyber Security Engineering Department, George Mason University, Fairfax, Virginia, United States

## Abstract

Phishing is a social engineering tactic that utilizes email and other forms of electronic communication that a malicious actor can use to trick people into giving their personal information, login credentials, bank account information, etc. Phishing is the number one cause of data breaches.

Breaches that started with a phishing email:

- Operation Phish Phry ($1.5 million)
- Walter Stephan ($47 million)
- Target/FMS Scam (over $200 million, 110 million users, 41 credit cards)
- Ukrainian Power Grid Attack (a whole country)
- Facebook/Google (over $100 million from 2013 to 2015)
- FACC ($61 million)
- Crelan Bank ($75.8 million)
- Ubiquiti Networks ($46.7 million)

Since phishing poses an imminent threat to everyone, our senior design project is to simulate a phishing attack for George Mason University (GMU). This attack will target undergraduate students above the age of 18, and use emails that manipulate different psychological factors that may affect students' susceptibility to a phishing attack.

## Introduction

This research project is based on a study previously conducted by our customer, Dr. Kathryn Laskey, SEOR, using GMU faculty and staff as the test subject. The prior study focused on the susceptibility due to the demographics of the test audience whereas this study is adapted to see what aspects of a phishing email is more appealing to the subject to convince them to click on a link.

**Four major parts are essential for the success of this study:**

- Experiment design
- IRB approval
- Environment set up and testing
- Execute experiment and analyze de-identified data

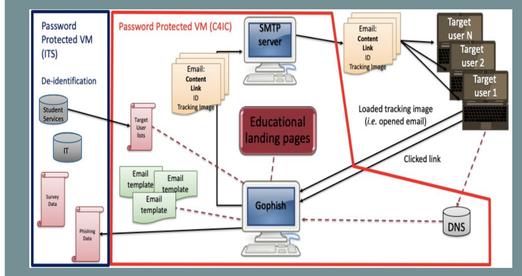
**Goals of this study:**

- Understand which psychological aspects will make a student more likely to click on a link.
- Seeing how many students will detect and report illegitimate emails.
- Determine whether incentive or punishment measure is more effective in combating phishing.
- Provide remediations to better educate students at GMU.

## Methodology

### Architecture Design

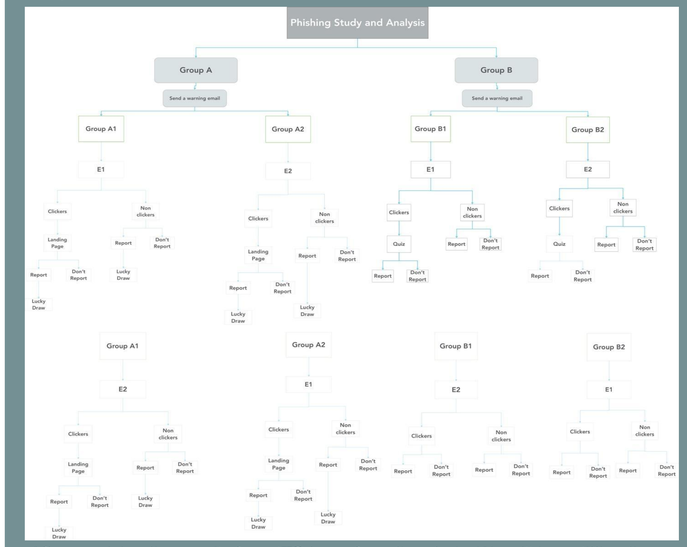
Mainly, the design outlines our methods and steps that must be followed to complete the project successfully.




This architecture demonstrates how each component interacts with each other via Gophish software.

- Email templates and landing pages are both loaded into the C4I server that hosts Gophish software.
- Gophish sends out the phishing emails to the students who are in the target user lists obtained from GMU ITS.
- Gophish automatically keeps track of who opened and clicked on the phishing emails.

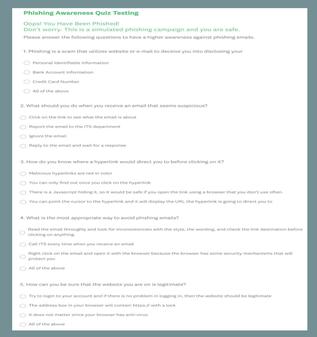
### Experiment Process Breakdown




After receiving the emails from ITS to perform the study. We will be splitting the emails into two main groups (Groups A & B).

- Group A & B will receive a general warning email from ITS about phishing email tips.
- We used 2 different email templates (E1 & E2) which are based on curiosity and fear psychological factors respectively.
- Each group will be divided into 2 subgroups that will receive each email template.
- In Group A, clickers and non-clickers will enter a lucky draw if and only if they report the phishing emails.
- The same procedure will also be repeated on Group B.
- But for Group B, clickers will receive a quiz to educate them about phishing scams to enhance their awareness about phishing.
- During the second round, all of the things will be the same except we will switch the email templates for each sub group.

### Landing Pages


We have created two different landing pages for clickers from different groups:

- The landing page on the left informs clickers that they were a victim of a fake phishing attack (Group A).
- The second landing page is a quiz about phishing to raise user awareness about phishing attacks (Group B).

### Email Templates



Email templates are created based on our hypothesis:

- Hypothesis states that fear-based emails will have more clicks than curiosity-based emails.
- The first template (on the left) is based on fear and it is a notification about unpaid balance in student account.
- The second template (on the right) is based on curiosity and it is about a first time virtual career fair due to COVID-19 outbreak..
- Measure the effectiveness of each template after gathering data.

### Verification and Validation

- For verification, we had weekly stand up meetings with our customer to ensure the design was to her standards via Agile model.
- For validation, with the help of our SME, we set up the Gophish environment with all the templates. We then sent testing emails to ourselves to see how they looked on various platforms. Adjustments were made as needed.

## Results

For the results, we intended on looking at a few different factors:

- which group had more students click on the phishing link.
- which round of emails had more clicks on the phishing link.
- which template is more effective.
- which group would have more phishing reports to GMU ITS.

Due to restrictions from COVID-19, we were unable to send emails out to students and were unable to conduct the experiment. Without collecting data from our rounds of intended emails, there was no analysis of data to show results or validate and verify our hypotheses.

## Conclusion

From this experiment, we projected that:

- Group A would report more emails to GMU ITS.
- There would be less clicks in the second round versus the first round
- An email template based on fear is more likely to receive a click than the template based on curiosity.

Dealing with sensitive information, the experiment could only be done on campus. Since we were not able to conduct the experiment, we were not able to draw any concrete conclusion.

## References

[1] "The Dirty Dozen: The 12 Most Costly Phishing Attack Examples," *Hashed Out by The SSL Store™*, Jun. 07, 2019. https://www.thesslstore.com/blog/the-dirty-dozen-the-12-most-costly-phishing-attack-examples/ (accessed Apr. 13, 2020).

[2] Brad, "The Top 5 Phishing Scams in History – What You Need to Know," *PhishProtection.com*, Jul. 24, 2018. https://www.phishprotection.com/blog/the-top-5-phishing-scams-in-history-what-you-need-to-know/ (accessed Apr. 13, 2020).

## Acknowledgements

Our team would like to express many thanks and gratitude to Dr. Kathryn Laskey for leading our senior design project, James Lee for helping us through the technical aspect of our project, and Professor Manzo for guiding us through the senior design process. Even though we have experienced some difficulties along the way, but we are still very grateful for having this as an opportunity to collaborate as a group and gain some insight on what does an industrial project looks like.

27 April 2020

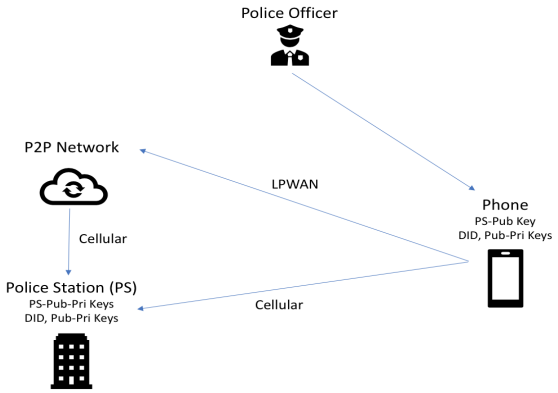
# Connection Resilient Bodycam with Built-In Non-Repudiation and Verification Features

Mattias Duffy, Prayatna Timalsina, and Rahma Moalin-Mohamed
Engineering Department, George Mason University
Special Acknowledgements to Steve D. Plessis & Tammer Olibah

## OVERVIEW

- An application for mobile devices to replace traditional **BodyCams** for Law Enforcement Officers
- Provides non-repudiation through hashing
- Multi-video streaming modes
- Peer-to-Peer Connection (P2P) via Apple's Multi Peer Connectivity Framework
- Sends captured video's metadata (Time, Location, Weather) to a server and constantly updates the status of the police officer

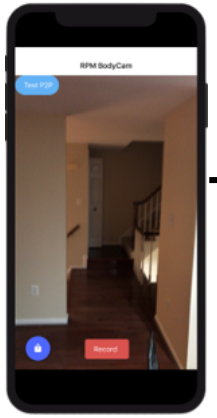
## METHODOLOGY




- Application constantly updates its status to the server every 5 seconds
- The captured video's data (hash, location, time) sent back to the server via Cellular network
- Automatically connects to nearby Peer-to-Peer (P2P) connection when cellular network is down

## RESULTS

Phone

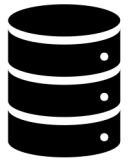


Metadata Packet
{
Time of Creation,
Video Hash,
Location,
....
}


Signed Metadata:

- Verifies sender
- Validates Video Authenticity
- Adds Non-Repudiation

Police Station
Video + Audit Log Server



Video Uploaded Manually Can be Cross-Checked with Metadata sent over the Network

## CONCLUSION

Our app demonstrates that it is possible to create a log of actions taken with the body camera and verify that recoded video is submitted un-altered without having to send full video data over a wireless connection. As connection quality is variable during operation, our app is also able to change data rates and connection methods to work in the challenging environments a police officer might face.

## ACKNOWLEDGEMENTS

We would like to thank both our Sponsor and Project Mentor for making this project possible.

## REFERENCES

1. Guidance and Support Benefits and Opportunities for Police Using Body Worn Cameras. (n.d.). from https://www.wcctv.com/benefits-and-opportunities-for-police-using-body-worn-cameras

2. Chapman, B. (n.d.). Body-Worn Cameras: What the Evidence Tells Us. Retrieved from https://nij.oip.gov/topics/articles/body-worn-cameras-what-evidence-tells-us

# Cyber Attack Map

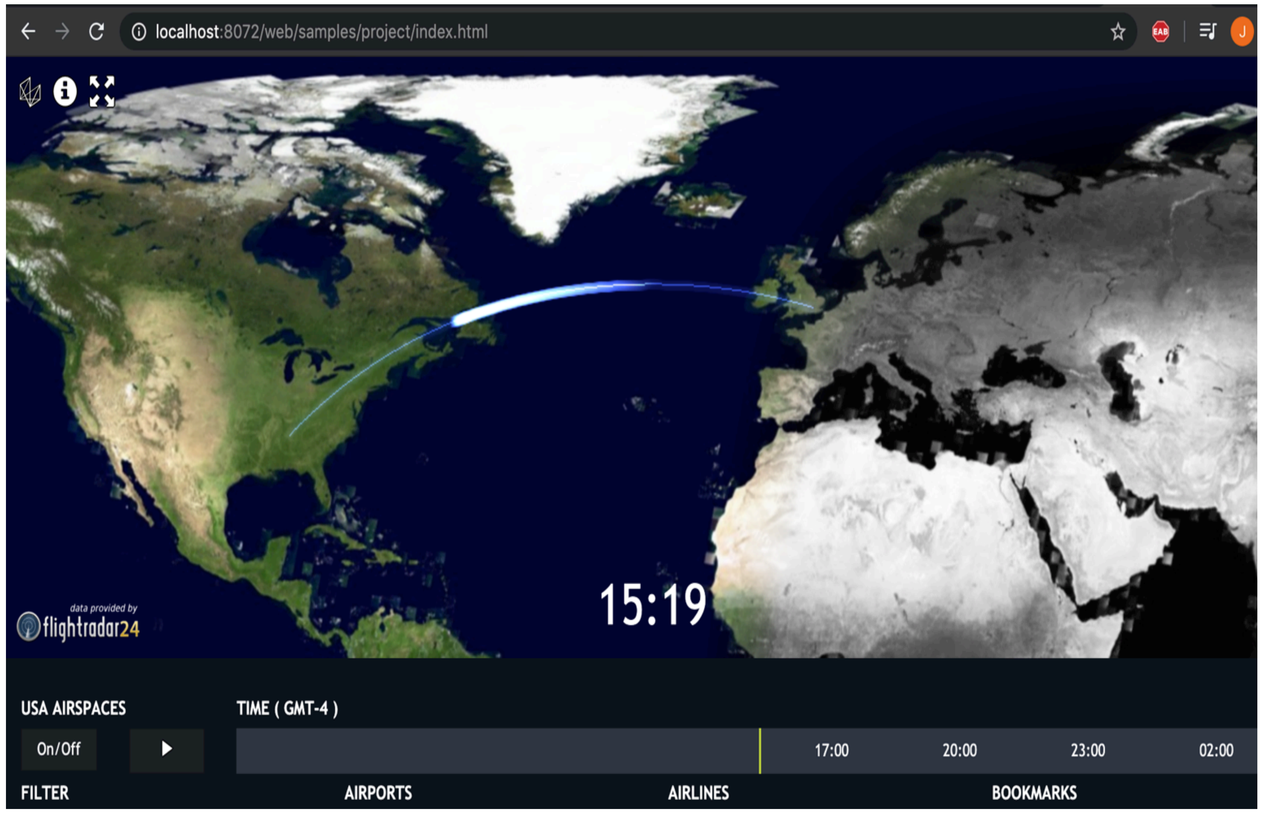
Warrie Proffitt, Hebah Beg, Jordan Poole
George Mason University
Sponsored by Hexagon Geospatial

## Introduction

With the world becoming more globally and virtually connected, the need for cyber security is continuously growing. While firewalls and intrusion prevention systems provide network protection, a solution is needed for security analysts to understand where malicious traffic originates. Our team has developed a Cyber Attack Map (CAM) that visualizes the source and target of malicious traffic in order to provide a tool for visual analysis.

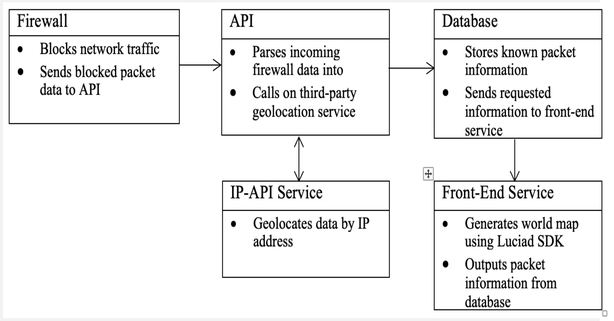
## Method

- This CAM utilizes LuciadRIA to visualize potentially malicious IP traffic gathered from the firewall. [1]
- For IP address geolocation, the CAM uses IP-API, a third-party service. [2]
- A database abstraction API connects the SQL database and the front-end service, providing easy-to-use API calls.




## Main Points

- The Cyber Attack Map provides a visual representation of incoming IP traffic that has been denied by the firewall. By mapping the source of the denied traffic, the Cyber Attack Map provides a visual analysis tool for determining the source of potential malicious traffic. This tool will be used to form a better understanding of where rejected IP traffic forms along with the rules used to deny the traffic.
- In order to properly interface with LuciadRIA, a geospatial visualization tool provided by the sponsors, the front-end of the CAM has been written in Javascript. The team has elected to use a SQLite database for persistent storage, utilizing a database abstraction API written in Node.js. Additionally, in order to properly identify the location of IP addresses, a third-party service has been integrated through asynchronous API calls.
- The figure on the right side displays the CAM data flow.




## Conclusion

- In conclusion this project provides Hexagon U.S. Federal a visualization tool for analyzing potential malicious activity. With the use of LuciadRIA, our team was able to build an efficient, lightweight tool to identify the source of suspicious network activity. As information is transported all around the globe through network packets, these packets of information provide information about the sender and receiver. The project presented to Cyber Security senior design students allowed for students to analyze the packets that are hitting Hexagon, the sponsor's firewall that maps the location that the packets are coming from in real time. The firewall has to protect a network from a malicious packet.

## Contact Information

- Warrie Proffit
  - wproffi2@gmu.edu
- Jordan Poole
  - jpoole6@masonlive.gmu.edu
- Hebah Beg
  - hbeg@masonlive.gmu.edu

## Resources

1. https://www.hexagongeospatial.com/products/luciad-portfolio/luciadria
2. https://ip-api.com/

## Acknowledgments

Our team would like to thank our SME Braden Pierson who helped us in our design. Thank you Dr. Powell for being our mentor and guiding us throughout this process. We would also like to thank our Customer Steve du Plessis for the opportunity to design this project.

# Virtual Honeypots for Infusion Pump Systems

Client & SME: Matthew Wilkes, INOVA

Anthony Tate, David Nguyen, Randy Maysaud, Ronan Roque, Salma Almaz

*Volgenau School of Engineering, Cyber Security Engineering, George Mason University*

## INTRODUCTION

- Infusion pumps are medical devices used to deliver fluids to a patient's body
- Previously standalone devices, now communicate in the network
- The increased connectivity also increased the attack surface for cyber attacks
- There is a need to defend these medical devices
- One form of defense are *honeypots* which are used to divert attackers from harming the legitimate systems


Figure 1: Baxter SIGMA Spectrum Infusion Pump

## APPROACH

- Developed honeypot to mimic Baxter SIGMA Spectrum Infusion Pump
- Uses open source software with minimal cost to the organization
- Pre-existing vulnerabilities such as weak authentication and hard-coded passwords were previously found on the infusion pump
- Raspberry Pi placed as a server on INOVA's network to deploy virtual honeypots
- Information generated from the honeypots will be logged and sent to processing by the Risk Management team
- After successful installation, the organization will be able to deter attackers from legitimate systems and identify malicious behavior on their network

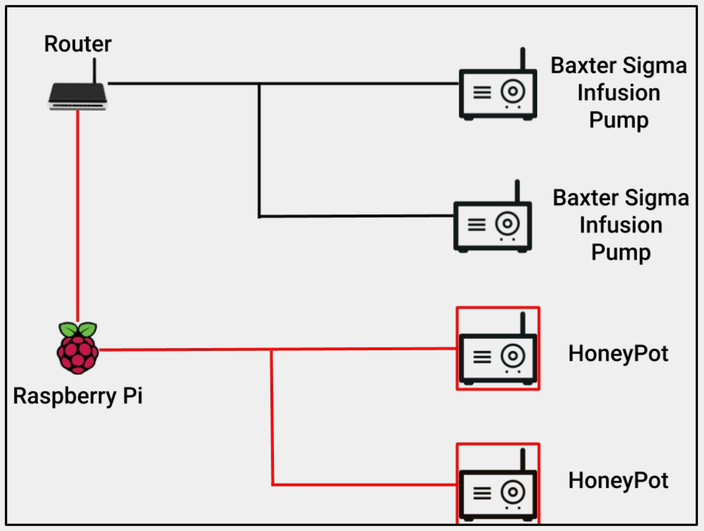



Figure 2: Virtual Honeypot Deployment

## METHODOLOGY

- Create a virtual honeypot server that can automatically inject virtual Baxter SIGMA Spectrum Infusion Pump honeypots into the network.
- HoneyD is the software used to create the honeypot server,
  - the software will be installed in a device that will act as the head of the server.
  - The device that holds the software will be able to configure and deploy the virtual honeypots, as well as host any information gained from the honeypots.
- Set up a raspberry pi as a host device:
  - The Raspberry Pi will hold the Honeyd software and control all the virtual honeypots that they release into the network. To release the virtual honeypots the Raspberry Pi must be connected to the router/switch of the network.
  - A useful aspect of Honeyd is that it has log feature, if an actor accesses one of the virtual honeypots it will log the attacker's actions and send the log back to the host which will be the Raspberry Pi.
- Deployment of the virtual honeypot:
  - The Raspberry Pi must be connected on the existing network,
  - The design will relies on a Raspberry Pi running the Honeyd software that will generate network traffic between the virtual honeypot and the infusion pump to emulate connectivity
  - The virtual honeypot is not physically connected to the network but the traffic is being routed by the software running on the Raspberry Pi.
  - That allows the administrator to shut down the virtual honeypot with ease and have a central device that holds all the logs that the honeypot generates which can then be analyzed by the security team.


Figure 3: Honeypot HTTP Service

```
--MARK--,"Fri 10 Apr 2020 03:51:04 PM HDT","apache/HTTP","","",,,
"GET /favicon.ico HTTP/1.1
Host: 192.168.1.75
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/80.0.3987.162 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://192.168.1.75/
```

Figure 4: Honeypot HTTP Service Log

## RESULTS

- In this project we implemented the methodology using a Raspberry Pi with HoneyD on a home network. On the home network we tested the information logging of the virtual honeypots, by breaching into the honeypot and matching the logs with the commands inputted. With the success of the proof of concept, it displays that this method works, a positive aspect of HoneyD is that it is incredibly scalable.
- Our Honeypot represent a Linux 2.2.13 (SuSE 6.3) and four open ports that each represent and log a specific service, reference Figure 3,4, and 5.

```
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
23/tcp open  telnet
80/tcp open  http
MAC Address: 00:00:24:1B:A4:86 (Connect AS)
Device type: general purpose
Running: Linux 2.2.X
OS CPE: cpe:/o:linux:linux_kernel:2.2.13
OS details: Linux 2.2.13 (SuSE 6.3)
```

Figure 5: Honeypot Ports and OS

## CONCLUSION

- This project aimed to develop a virtual honeypot solution for the Baxter SIGMA Spectrum Infusion Pump
- Virtual honeypots were deployed using Honeyd
  - Allows for easy configuration and management of the honeypots
- Although testing was done on a Raspberry Pi, the software can be run on heavier hardware
  - Solution is very scalable
- Currently the solution is just for the Baxter SIGMA Spectrum Infusion Pump, but it can be easily configurable to replicate other devices

## ACKNOWLEDGEMENTS

We would like to thank INOVA for sponsoring this project and especially Matthew Wilkes who has supported our team throughout this project. Furthermore, we would also like to thank Dr. Manzo for mentoring our team.

## REFERENCES

[1]https://github.com/DataSoft/Honeyd
[2]https://www.baxter.com/sites/g/files/ebysai746/files/styles/portrait_image/public/2018-06/spectrum-iq-infusion-system.png?itok=KkmIHF1A
[3]Mahajan, S. (2020). Intrusion Detection System Using Raspberry PI Honeypot in Network Security. [online] Pdfs.semanticscholar.org. Available at: https://pdfs.semanticscholar.org/9c2e/b0a9817be134c28c73c24a73600dd1cd15b8.pdf [Accessed 3 Mar. 2020].
[4]http://www.hackinsight.org/news,90.html

April 27, 2020

# INOVA® Biomedical Honeypot Device

GEORGE MASON UNIVERSITY

## Introduction

**Task:** Our team worked with INOVA to create a biomedical honeypot to collect data for information security analysis.

**Goal:** To attract threat actors so we can understand their tactics, techniques and procedures used when attacking the device.

### What is a Honeypot?

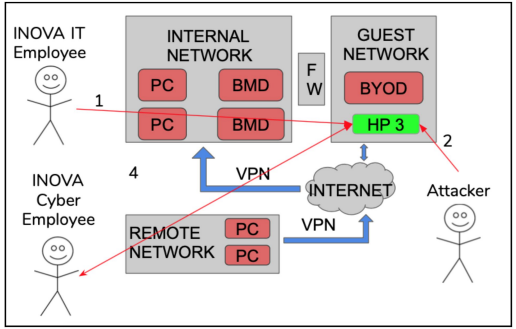
Honeypots are designed to resemble valid systems or devices, and appear to be a legitimate part of the network they are in; they are used to collect information about their attackers.

## Our Implementation:

We created a two machine system. One was a logging server and one was a simulator. We rerouted all logs and commands from the simulator to the logging server.

## The Code:

We used python to create a simulation script. We also got a basic Linux Enumeration script in python to run on our system to simulate an attacker.




**application_name**
SIM-LOG

**facility**
local0

**level**
6

**message**
INFO: 04-04-2020 09:16:30 AM {'KERNEL': {'msg': 'Kernel', 'cmd': 'cat /proc/version', 'results': ['Linux version 4.19.97-v7+ (dom@ buildbot) (gcc version 4.9.3 (crosstool-NG crosstool-ng-1.22.0-88 -g8460611)) #1294 SMP Thu Jan 30 13:15:58 GMT 2020', '']}, 'HOSTN AME': {'msg': 'Hostname', 'cmd': 'hostname', 'results': ['Hogsmea d', '']}, 'OS': {'msg': 'Operating System', 'cmd': 'cat /etc/issu e', 'results': ['Raspbian GNU/Linux 10 \\n \\l', '', '']}}

**source**
Hogsmead

**timestamp**
2020-04-04 13:20:14.874 +00:00

## Methods and Results

Our system was tested with an "attacker" script that is a open-source enumeration script. It aims to connect with our system and query for more information. When this script is run, it triggers our logging server and alerts are sent to a designated party.
By running our script and comparing its actions with recordings on our logging server, we verified the accuracy of the model.

## Conclusions and Future Work

An attacker, when trying to compromise a seemingly vulnerable device, will try a wide variety of tactics including: testing default passwords, accessing open ports, scanning for vulnerable sensitive data, and much more.
By constantly monitoring and securing a device, when an attacker performs these actions, increasing the security and response time of the device will become increasingly easier and more efficient, as well as predicting and preparing for future attacks.

### How can an organization use this product?

Our model is flexible enough to be applied to a wide variety of products and situations. By taking our model and adding device specific information, an organization will have an easy, ready-to-go reusable honeypot and logging server that will help them understand what threats they are facing and increase their overall security posture.


### Acknowledgements

We would like to thank our advisor Thomas Winson, our professor Peggy Brouse, and our INOVA Subject Matter Expert Matthew Wilkes.


### References


https://github.com/sleventyeleven/linuxprivchecker/blob/master/linuxprivchecker.py

# Using Generative Adversarial Networks to Produce Synthetic Overhead Imagery


## Cyber Security Engineering

Bawer Alissa, Twinkle Gera, Amir Itayem, Adalid Helguero, and Mohammad Saad

Sponsors: Tim Parker and Jonathan Brant


## Background

- **Generative Adversarial Networks (GAN):** are neural networks in machine learning which are composed of a discriminar and a generator.
  - These two model are train against one another with the rules of a zero sum game, where a loser and winner is always involved
  - Progressive GANs and Deep Convolutional GANs are two different types of architecture in producing synthetic images
  - These two different architecture involve the use of convolutional neural networks.
  - In order to produce the best synthetic images, both were tested.
- **Convolutional Neural Network (CNN):** are neural networks specifically designed to take images as input to recognize edge detection and any other image details
- **Deep Convolutional GAN:** DCGAN is a design architecture for GANs which also uses transpose convolutional layering and strides in order to upsample and downsample the input
- **Progressive GAN:** One of the latest design of a GAN architecture and builds of a DCGAN. A progressive GAN downsample an image to a small size and trains until sufficient learning takes places. It then increase the image dimensions and repeats the same steps again until the desire image shape is produce.
  - Progressive GANs are used to help produce images of higher resolution because it increases the stability of the models.

## Objectives and Materials

**The objective of this project was to produce a realistic synthetic image using a DCGAN and/or a Progrossive GAN**

**Materials**

- **Tensorflow:** is a deep learning frankwork for machine learning.that was used.
- **Keras:** is a high API that uses Tensorflow as a backend. Keras was used because of the simplicity and the intuitive in training models
- **What is xView?:** xView is one of the largest public datasets of overhead images. These images that are taken from around the world contains complex scenes. xView believes on progressing learning efficiency, improve detection of fine-grained classes, and reduce minimum resolution.
- **How large are the overhead images?:** The overhead images usually are 2900 x 3100. However, for our project those images have very large dimensions. This could be beneficial for the accuracy of machine learning, but having a dataset will cut down the training rate drastically. Our solution was adding a chipping function that is utilized to downsample the images into smaller image of the size of 256 by 256 pixels.

## Generator & Discriminator

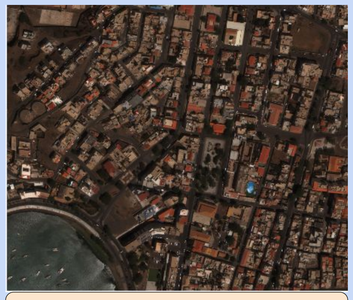
**Generator:** The generator is integrated into the GAN. First the generator will build images using the architecture. It will start from a 100 dimensional noise vector. The first layer adds a fully connected layer called “dense” into the generator architecture. The desired image size is 256x256x3 pixels. The dimensions consist of the length and width of the image. The last dimension shows if it’s in color or grayed scale

**Discriminator:** The discriminator inspects the images produced by the generator and determines whether the images are authentic or synthetic. These results are provided back to the generator, which uses this data to make synthetic imagery more realistic until it can deceive the discriminator. The discriminator penalizes the generator if it produces implausible results. The discriminator then takes an image and applies convolutional layers until a one dimension noise vector is left. Finally, the discriminator uses the vector to predict whether the image is synthetic or real

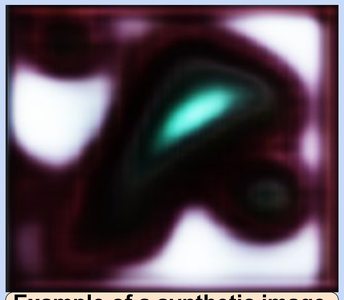
## Training Method

**DCGAN_**- When the DCGAN start training, it first sends a random noise vector of 100 randomly generated image samples to the generator. Afterwards, a variable is created which contains the image data of fake and real images. While training, we used a batch size of 100. This batch contains 100 randomly selected images from our entire chipped dataset. These images are what help both of our models to learn how to generate synthetic images. We sample 100 random images 22 times and this would be one epoch. We train the DCGAN usually for 50 epochs.

**Progressive GAN** - The progressive GAN starts by convoluting an 8x8 image for the first few epochs. Both, the generator and discriminator, will upsample up to 16x16 and so on during training based on the epoch. A layer is added and the image size is upsampled repeatedly until 256x256 size images are being generated. At this point, we continue training until we reach around 50 to 70 epochs.

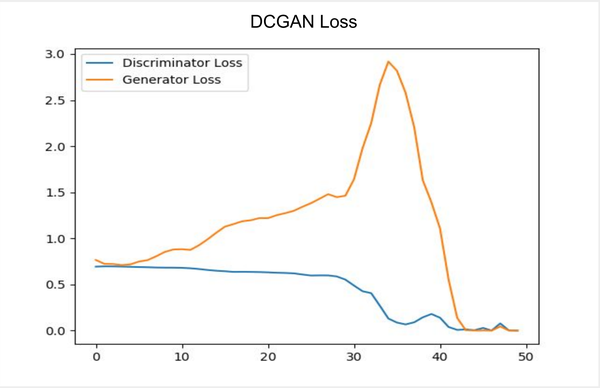


**Example of an xView image**

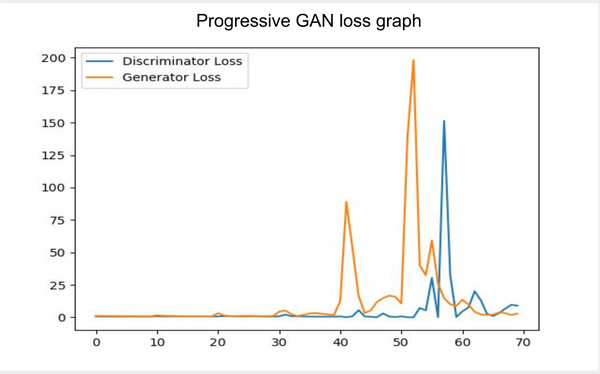


**Example of a synthetic image during training**

## Loss Graphs




Graph above depicts the binary loss of our DCGAN. This measured the loss for each epoch using a binary cross-entropy function for 50 epochs.




Graph above depicts the binary loss of our Progressive GAN. This measured the loss for each epoch using a binary cross-entropy function for 60 epochs.

## References


msaad4@gmu.edu ahelguer@gmu.edu aitayem@gmu.edu tgera@gmu.edu

balissa@gmu.edu jonathan.c.brant@lmco.com tim.parker@lmco.com

# Finding Zero-Day Vulnerabilities

Fletcher Davis, Peyton Edmondson, Austin Griffith, Austin Harlow, Clifford Krey
Department of Cybersecurity Engineering, George Mason University

## Objectives

Assess the attack surface and perform vulnerability research on the embedded software system of the Huawei HG8245H router.

Help Sponsor better understand vulnerabilities in devices to aid in supply chain procurement risks.


## Open-Source Tools

**Kali Linux**
Operating system used for our research, development, testing, and virtual exploitation environment.

**Burp Suite**
Analyze network protocols between router and client

**Firmware Slap**
Used on the router's firmware to find potential buffer overflow vulnerabilities

**Huawei Forums**
Download router firmware and discover super-admin credentials

## Vulnerability Research


**Open-Source Research**
Our initial focus was understanding the HG8245h (right). Then metadata was gathered and used to perform a search. Huawei forums were especially helpful.

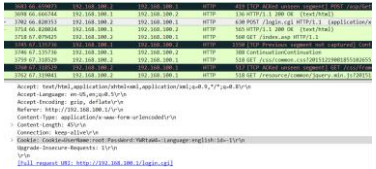
**Firmware Extraction**
In order to find software vulnerabilities on the router, we must obtain a copy of the firmware. Through the debug interfaces UART and JTAG, as well as through the vendor image repositories, we were able to obtain a binary image.

**Firmware Analysis**
Firmware was acquired online and pulled from device, Then the filesystem was extracted and examined using Binwalk. Lastly, Firmware Slap was used on the router's firmware /bin/ directory to find buffer overflows.

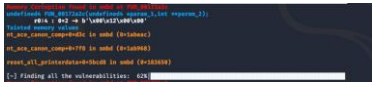
**Hardware Analysis**
Router was disassembled and internal components were photographed and documented. Data for chipsets was collected.

**Software Analysis**
List of software pre-installed on the router was cross referenced with known vulnerabilities for each software's specific version.

**Penetration Testing**
Used open-source tooling to analyze the web interface for web application vulnerabilities. Utilized tooling to fuzz UPnP input on the exposed UPnP port in an attempt to cause a crash.



Wireshark output of the wireless password disclosure vulnerability

Firmware Slap: Automated firmware analysis tool scanning firmware function for vulnerabilities

## Results

**Firmware Vulnerabilities**
333 potential buffer overflows within the router's firmware could be exploited to allow an unauthorized access of memory.

**Hardware Vulnerabilities**
Exposed debug interfaces allow access to filesystem to extract device memory and firmware. Unencrypted firmware package. Lack of integrity and authenticity check allowed for the upload of malicious firmware that included a backdoor.

**Web-Based Vulnerabilities**
Wireless HTTP router credential sniffing vulnerability found a web cookie that contains unencrypted login credentials. Default super-admin credentials were found, contacting Huawei is required to change them.
Username: telecomadmin Password: admintelecom

## Conclusion

Vulnerability research on the Huawei HG8245H router yielded numerous vulnerabilities and even more potential vulnerabilities

With additional time the workflow can be continued to reverse engineer the potential vulnerabilities found

The results of this research validate the United States Government's mistrust in Huawei

Workflow can be recreated on other devices to test their security

## Acknowledgments

Our wonderful project would not have been possible without Professor Rock Sabetto for organizing this Capstone class, our mentor Dr. Powell for his guidance, Sasha Morgan for representing our sponsor Lockheed Martin, Dr. Jones and Dr. Winston for guidance on reverse engineering, and Chris Roberts for creating "Firmware Slap".

## References

C. Roberts. "Firmware Slap." Github. https://github.com/ChrisTheCoolHut/Firmware_Slap (accessed April 5, 2020).
U.S. House. 115th Congress, 1st session. (2018, Apr. 13). *H.R.5515, John S. McCain National Defense Act for Fiscal Year 2019*. [Online]. Available: https://www.congress.gov/bill/115th-congress/house-bill/5515/text
D. Burstein. "Huawei's Very Red World." Circle ID. http://www.circleid.com/posts/20190830_huaweis_very_red_world/ (accessed April 5, 2020).

# Windows Packer | Loader

**Andrew Chapin** (achapin2@gmu.edu), **Andre Herrera** (aherre12@gmu.edu),
**Carl Bai** (cbai2@gmu.edu), **Hunter Rowlette** (hrowlet2@gmu.edu),
**Mitch Palmer** (mpalmer7@gmu.edu)
Allision Elfring, Lockheed Martin
School of Cybersecurity Engineering, George Mason University

Team BluSh3ll

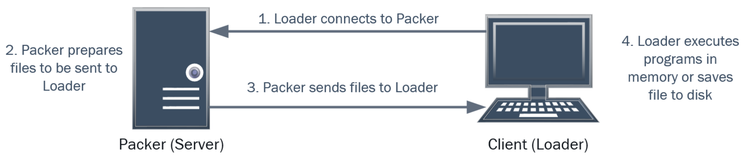
## Background

Our project fulfills a request to produce a software toolkit that allows for remote code execution completely in RAM and file transfer via a service running on a remote host. The goal of our stakeholder, Lockheed Martin Corp. (LM), is for our research to identify a unique way to accomplish this task.

Packers are generally used to encrypt, compress, and sometimes obfuscate executables/files that are fed to them. Our packer also connects to a loader for remote delivery of these files.

Loaders read program instructions into memory. They are designed to be small and reliable with minimal functionality are often difficult to detect.

Our deployment of a loader is as follows:
1. Loader is deployed on a client system and connects back to server (C&C)
2. Packer compress and encrypts files to be sent to Loader
3. Packer send files to Loader
4. Loader, on the client system, executes PE files in memory or saves the files to disk.



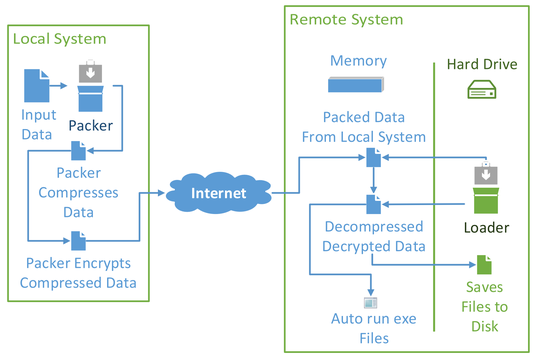

This project is useful for discrete delivery and execution of files to remote systems.

LM Requirements:

1. The toolkit must be comprised of two separate executables – a “packer” and a “loader”
2. The “packer” runs locally on Linux, compresses, then encrypts with AES via a user-provided password before sending data to remote hosts
3. The “loader” runs on a Windows remote host as a service, receives incoming packed data, decrypts/ decompresses, and executes any PE files entirely in RAM (i.e. without touching disk). Other loader operating systems were desired.

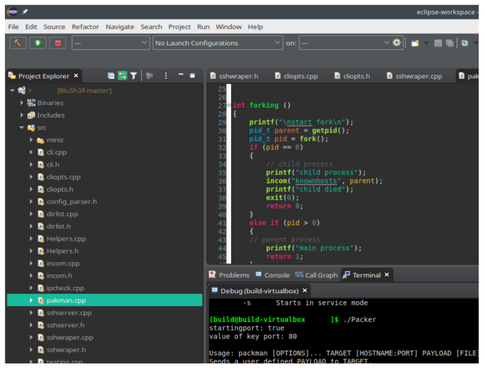
## Design Overview

In the planned design, the packer program on a Linux system receives files from a user. The packer takes this arbitrary data, compresses the data, and then encrypts the data. These "payloads" are stored on the system in a single file. Then, a user can select a payload to send to known hosts containing a loader. The user can choose one or more hosts. The payload is sent over the internet. On the remote Windows host, the loader is running as a Windows service. It waits on a preconfigured port for incoming data. When it receives a payload, it will decrypt and decompress the data entirely in memory. Then, depending on user-defined, configurable conditions it will either execute the program completely in memory as a separate thread/process or save the file to the disk.




## Implementation

Both tools were written in the C++ programming language. We choose C++ because it allowed us to perform low-level memory management and it contained the functionality via standard or external libraries to achieve the requirements set forth. Also, by writing both the Packer and Loader in the same programming language, we were able to duplicate effort (e.g. the encryption function could be easily ported into the decryption function for the loader). For our development environment, we utilized a Manjaro Virtual Machine (VM) running Eclipse to code the Packer and part of the Loader development. By sharing this VM, our development team was able to ensure that the code would build and compile without having to worry about "it worked on my system" problems. For most of the loader development, we utilized Microsoft Visual Studio – since the loader was meant to run on Windows Operating Systems (OS).



*A screenshot of the Manjaro development environment.*

We utilized external C++ libraries that supported our needs for compression (miniz library – which implements zlib (RFC 1950) and Deflate (RFC 1951) compressed data format specification standards); AES-256 encryption (openSSL library) and networking (libSSH and libSSL libraries). To communicate with the loader, the SSH protocol was utilized. We choose SSH for our design because it handled the management of encryption keys reducing our development requirements for a keystore.

Our packer was designed to accept both command line arguments and direct user interface input. The unistd.h library is used with the getopt() function to parse through passed arguments and set key options flags in the program. If the program is run with improper arguments or the “-h” flag, the help page will be displayed. Once all command line arguments are properly parsed, the program calls the CLI code which presents the user with an interactive command line interface.

```
Usage: packman [OPTIONS]... TARGET [HOSTNAME:PORT] PAYLOAD [FILE]
Sends a user defined PAYLOAD to TARGET.
Example: packman 192.168.0.1:1337 ~/payload.exe

        -h, --help      Prints this Help Page
        -s      Starts in service mode
```

*A screenshot of our help page.*

The loader was also written in C++ so we did not have to duplicate effort (e.g. once compression/encryption was coded, the decompression/decryption could be ported with minor changes). Also, C++ allowed us to utilize the .NET framework in Windows to interact with the operating system (OS). It also performed code execution entirely in memory.

```
[build@build-virtualbox Debug]$ ./Packer -s
Starting

 Choose an option:
1. List Available Payloads
2. Add new payload
3. Send payload
0. Exit
> 1
current payloads

0 payload.txt

 Choose an option:
1. List Available Payloads
2. Add new payload
3. Send payload
0. Exit
> 3
```

*A screenshot of our home page command line interface in action.*

## Results

A delivered product! Our packer and loader tool combo was delivered to our stakeholder. Our product helped enable their mission and objectives. We also provided the research we conducted along with the product. As our assignment was both to conduct the research into the technology and develop the product, both we equally valuable. We found that R&D of such a product is quite difficult, especially in the area of executing code completely in memory. We researched products that delivered similar solutions to ours. From what we found, most only delivered a portion of our entire system. E.g. one product advertises receiving data from remote systems and another specializes in code execution in memory. Our product contained a lot of subsystems. By researching the best methods specialized programs succeeded in building these subsystems, we were able to aggregate a successful full product.

We met the required spec from our stakeholders. Very few of the desired objectives were met though, due to constrains such as a quick timeline and barriers in development

Lessons Learned
- A standard development environment is essential
- Test cases make development easier and quicker
- Time management is essential (both for development and the business end of assignments)

| Function | Technology | Description |
|---|---|---|
| Code Base | C++ | Fast, lightweight, & cross-platform compatible. |
| Compression | miniz | Single C source file zlib-replacement library. |
| Encryption | openssl | General-purpose cryptography library. |
| File Transfer | libssh | Multiplatform C library implementing the SSHv2 protocol. |
| File Handling | FS Library | As of C++17 filesystem library. |
| Argument Handling | getopts | Parses the command-line arguments from *argc* and *argv* |
| Process Handling | Fork | Creates a new process by duplicating the calling process. |

## References

[1] “libssh 0.9.3,” libssh. [Online]. Available: http://api.libssh.org/stable/. [Accessed: 16-Mar-2020].
[2] R. Geldreich, “richgel999/miniz,” *Github*, 09-Mar-2020. [Online]. Available: https://github.com/richgel999/miniz. [Accessed: 16-Mar-2020].
[3] “Filesystem library,” *cppreference.com*, 15-Jun-2018. [Online]. Available: https://en.cppreference.com/w/cpp/experimental/fs. [Accessed: 16-Mar-2020].
[4] “Payload Types in the Metasploit Framework,” PAYLOAD TYPES IN THE METASPLOIT FRAMEWORK. [Online]. Available: https://www.offensive-security.com/metasploit-unleashed/payload-types/. [Accessed: 16-Mar-2020].
[5] arno0x0x, “Windows oneliners to download remote payload and execute arbitrary code,” Windows oneliners to download remote payload and execute arbitrary code, 18-Apr-2018. [Online]. Available: https://arno0x0x.wordpress.com/2017/11/20/windows-oneliners-to-download-remote-payload-and-execute-arbitrary-code/. [Accessed: 16-Mar-2020].

## Acknowledgements

We would like to thank Professor Sabetto for his constant words of support and encouragement, believing in us when we didn’t believe in ourselves.

We would also like to thank Allision Elfring for her consistent advice and her invaluable comments which helped us to constantly improve our project.