Project Nighthawk - Virtual Reality Visualization and Interaction with Automated Cyber Attack Tool

By Wyly Andrews and Ruben Tipparach

Introduction

Abstract

Development of cyber attack and defense applications have enable end users that have a deep understanding of the concepts behind cyber security to execute code to perform various exploits. In a lot of cases, the cyber security firms employ engineers that develop and test applications within a contained environment, and then stand behind military personnel instructing them on how to execute the code. The experts often do not have the clearance to deploy cyber attacks themselves, and often have to direct commanding officers or cyber warfare personnel to carry out these attacks. Thus, there is a need for cyber security software that can scan and analyze a network for vulnerabilities and carry out the attack by non-experts in a visual and intuitive manner.

Project Nighthawk proposes the solution of integrating software designed to scan for vulnerabilities and demonstrate the ability to deploy exploits. This report will go over an in depth analysis of the problems and solutions required to create this tool for cyber warfare analysts and security experts to build on top of. This report goes in detail the steps to carry out cyber attacks in the Kali operating system hacking into a Metasploit 2 server, the necessary communication API, and the representation of data required for non-cybersecurity experts to make the best possible decisions.

Thesis Questions

The main problem faced in developing Nighthawk are broken into two main components.

1. **How does a Virtual Reality environment allow the user to better visualize a network environment, and identify vulnerabilities and threats?** In previous research, there has been a lot of study in the analysis of data from a 2D perspective, although data can be represented in 3D via a standard computer monitor. This research, aims at finding a new framework for effectively representing cyber-systems in Virtual Reality using stereo 3D rendering and positional tracking of the head and hands which should improve the visibility of the data, and being able to navigate through the data within an immersive environment.

2. **How would VR enable the tools and interface required for users to interact with the tools to carry out cyberattacks in a real-time setting?** This question tackles the design and architecture of a virtual interface that communicates with the exploitation tools within Kali OS. The question here, seeks to find an interface and navigation system for moving through, searching the data, and interacting with the host(s). Similarly to the visualization aspect, the User Interface for Cybersystems in VR seeks to build an architecture for implementing an intuitive system for carrying out cyber attacks.

Approach

For the visualization problem, the research team looked at developing a few VR prototypes and create data mockups to simulate what actual data scanned from a network would look like. From here, the visuals were organized, and the data structures were tuned to reasonably match the output from the scan results of NMap.

As for the interface, the team looked into existing tools for virtual reality, and created user interfaces that surrounded the user in 3D virtual space to allow the user to quickly interact with each element within the environment. Applications like these in the future that utilize VR or AR (Augmented Reality) space will become more common, thus the adoption of existing tools, and improving upon standard non-VR User Interface and User experience designs would help ease them into a familiar but expanded user environment.

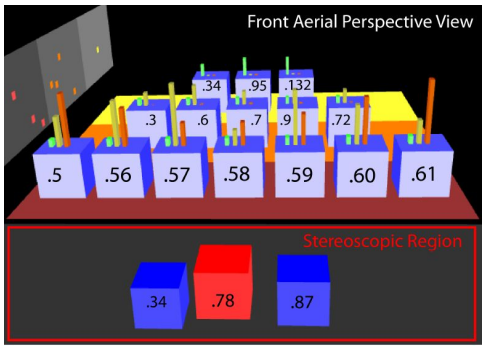
Previous Work

Automated Cyber Attack

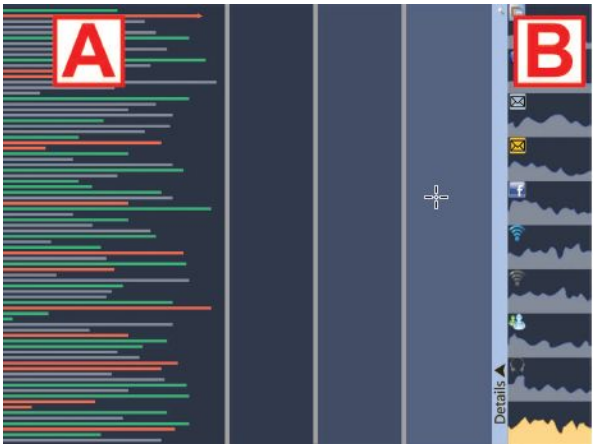
This project is an extension of a previous project known as EZhack [1], which utilizes the Kali OS and the tools bundled with it to initiate the scan and launch the exploit. This tool is written in Python, and uses a Remote-Procedure-Call to execute commands on the Metasploit framework on to the Metasploit 2 OS. This allows the tool to “automate” the process that a hacker would go through to scan a system and remotely execute code.

Data Visualization and User Interfaces

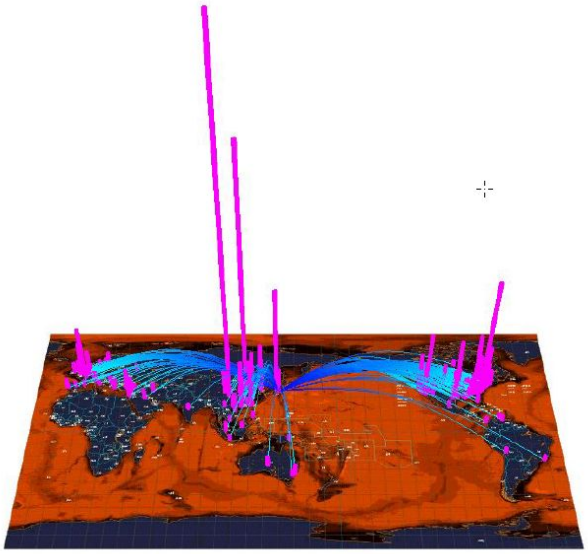
The team looked at several ways to analyze the data. One of the first approaches was to look at existing work that attempted to create easy-to-understand and intuitive visualizations of the network. Some ideas we found for visualization, and standards created by other researchers are as below.



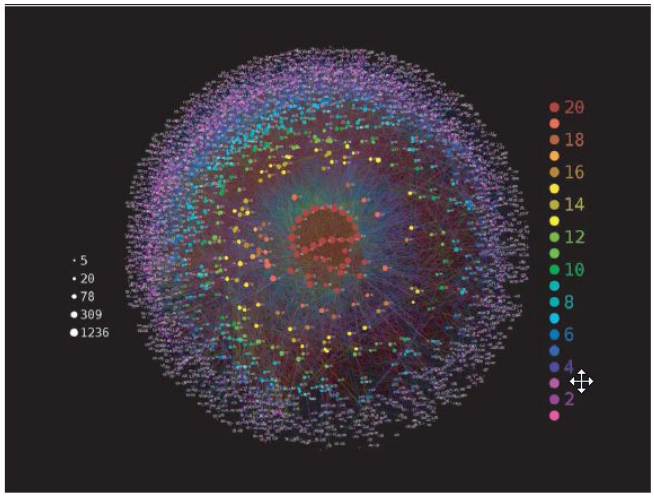
In this demonstration of 3DSVAT, the nodes on the bottom are stereoscopically integrated into an array of nodes above [2]. When they're vulnerability level increases, they become separated(using stereoscopic depth) so that the system administrator will be able to see separation of the stereoscopic nodes from the background nodes. In VR, stereoscopic images are inherently the nature of the medium, and thus an implementation using this 3D technique makes it a very useful.



In SEQViz similar nodes are grouped together on the left (A), and the services running, and network traffic they are using are shown in B [3]. A 3 dimensional view of network traffic could be overlayed in 3D showing in comparison which service sees most traffic. It could show vulnerability in an aggregate set of services.



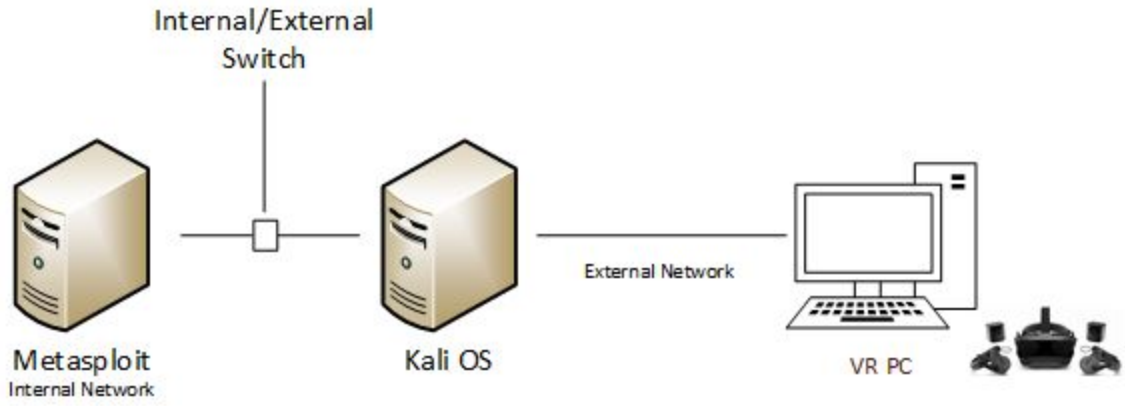
The STARMINE project visualizes nodes as geographic points [4]. Cyber attacks proposed in this project, Nighthawk, are designed to run on a remote host, thus visualizating the locations of nodes and their connections would make this a tool not just for carrying out attacks, but managing attacks on a global scale as well.



Although more data oriented than cyber-networks, LaNet-vi is a tool used to visualize large networks [5]. This tool allows the computation of each node's position within the network to cluster according to certain attributes on a scale. Data in this particular network are organized according to the number of connections they have within the network. Color coding these connections and segregating them into rings could show choke points within the network, thus signaling the most vulnerable points to the user. Thus, allowing them to disable and cut off a large portion of the network by attacking on the most interconnected nodes..

Overall these different design ideas have a few things in common, the network connections are vital to understanding relationship between the different nodes scanned on a network and the visualization of the classification of the state of each node - their “vulnerability” state. This helps the researchers understand where the vulnerabilities are and where the attacks happen, or are most likely to happen. This plays a huge roleon how to design a system that can effectively cripple or disable a target host or network.

Design



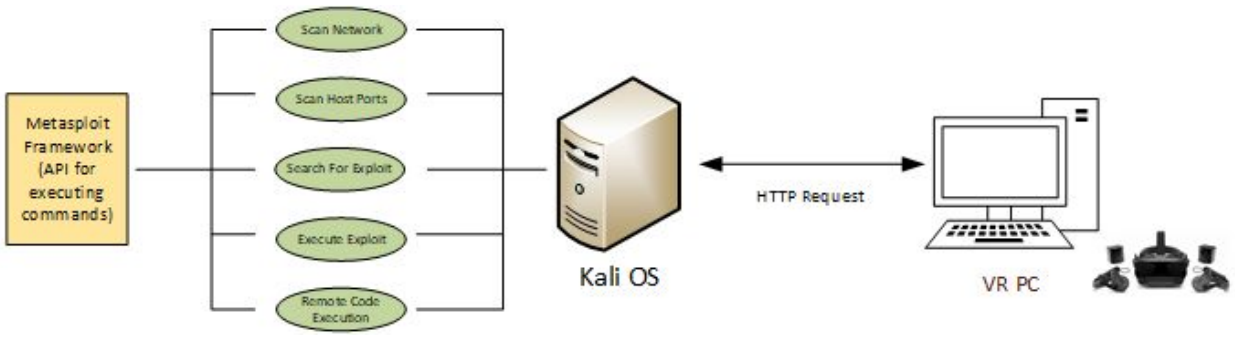
In the Cyber Attack Simulation System, a virtual machine hosting 2 linux servers are used to model the “attacker” and the “target” host. The Kali server acts as the “attacker” as it is a specialized operating system designed specifically for cyber security research, and thus containing a multitude of tools commonly used to search for vulnerabilities within a network. The “host” server uses Metasploit 2, a server designed to have various flaws that allow the Kali user to test different tools and methods on. Metasploit 2 in particular was chosen as it had the most benefit from the code base that this project was built up, which was the EZHack tool by Isaac Burton. EZHack does much of leg work for this project in terms building of a foundation for creating more exploits and remotely operating the Kali Server from a different OS or environment.

On the right side of the network is the virtual reality machine. This device hosts the virtual environment for the user to interact with. Interactions connected via the Kali OS will execute the data sweep on the local network and the Metasploit server. The scanning of IP Addresses targeted by the server will allow the user to have an overall view of each IP address and the vulnerabilities associated. These host servers will be tagged and highlighted appropriately. In the future, a geographic component will come into play and allow the user to even more finely visualize the (approximate, if available) geographic location of each host server. Finally, the the VR PC will choose the exploit they want to run and execute that exploit on the targeted servers. The VR system will also allow users to quickly react to changes on the server, such as falling to other exploits when one of them fails, or to allow manual entry in console mode to gather more information or execute other commands if the VR system fails to cover certain cases.

System Architecture

Overall the Nighthawk project utilizes two main components - the Kali Server and the User's VR Personal Computer. For the network architecture portion, the Python server originally designed to automate hacking tools will been converted into a RESTful API using the Flask library, a library designed specifically for hosting web services in Python. For the VR system, it will consist of an HTTP Client for the API, the Visualization section, and finally the UI control system.

Network API Design



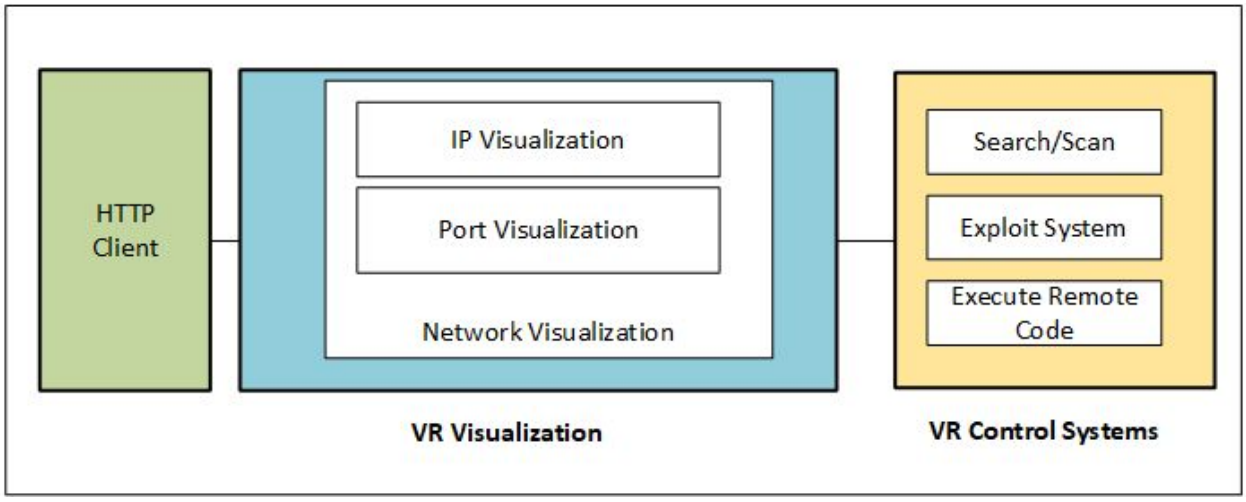
In the network design, the operations are split into 5 categories.

1. **Scan the Network.** This will take an IP Address, and scan any devices connected to it. For this project, the demo system is using the network where the code is hosted on, although in theory, this could be any gateway connected to the internet such as a router, domain controller, or just a node with in the server.
2. Once the server has finished scanning for potential host targets, the second phase of **scanning the port** begins. This, also in theory, can be automated to mass scan all the hosts within a given network, or a range of hosts available.
3. **Search for the exploit.** This step involves searching an internal database for exploits that can be run on the system. The desired exploits are usually exploits that enable some sort of code to run on the target host machine, such as a keylogger, a password capture tool, or the ability to inject code into an unpatched part of the host server.
4. **Execute the exploit.** This is the initial step to launching an "attack". When you exploit a system, you are technically stripping it of its defenses. In a lot of cases, this involves some kind of firewall, login page, or anti-malware system to prevent the system from being compromised. This system currently will not be field tested, or tested on any actual modern device, since the primary focus here is to create the VR system for execution, and not on the effectiveness of the execution itself. It will however be tested on a model system, the Metasploit Server.

5. **Remote Code Execution.** The final step involves actually attacking the system. This can be a virus, rootkit, or any kind of script that would prevent normal operations at the right time,as well as the ability to gather information to gain further levels of access within the network.

Although, not listed within the scope of this project, further methods of executions are planned. Such as the ability to erase the user tracks when they log out of the system maybe also be a feature in the future and even the ability to geographically query for host nodes.

Virtual Reality Control System Design



On the VR side, this subsystem consists of 3 main components.

1. The **web client** is for communicating with the Kali Server.
2. **VR visualization** is broken into two parts:
    a. IP Visualization - This will detail the overall network and its connections.
    b. Port Visualization - Also referred by our team as the application visualization, will allow the system to figure out which version each application is on, and allow for a possible vulnerability to be found
3. **VR Control System** This component is responsible for the operations of the system. Navigation through the data, such as searching and targeting hosts will be done through this component. Essentially, it is the VR user interface layer where the user will interact with the visualization model, and make decisions based on the data represented.

Implementation Part 1: Network Visualization

This section will describe the steps taken to implement the network visualization component of project Nighthawk.

Data Model

In order to implement a simulation, we needed to create a data model to test with, and allow the VR application to generate the visuals in real time. All visuals and informational text in the simulation are generated according to the data that is fed via custom scripting or scanned from an actual virtual network environment.

Simulation Model


```
 "data": [
  {
   "id": 0,
   "octets": [  37, 55, 125, 50  ],
   "macAddress": [ 165, 63, 182, 11, 111, 38  ],
   "status": 1,
   "latency": 252,
   "deviceType": "Broadband Router",
   "os": "Linksys Default",
   "connections": [
    2
   ],
   "ports": []
  }
}
```


The following snippet is an example of the data model used in the network simulation module, this is similar to the NMap data in the actual payload retrieved from the Kali OS server program, more on that in the actual model description. This simulated version was designed to quickly iterate the visualization component, since the Kali Server was not yet available for set up at this point in the project. The accuracy of this model is rather similar to the output from NMap, however it does not have a similar data structure overall. Parts that allowed the research team to test with were the IP addresses its connections (modeled as indices in an undirected graph). Although port, and a description data for each web service was also included in the model, this portion had not yet been integrated into the 3D visualization portion. Both this simulation model, and the actual data model are both compatible with the system, allowing the project to author customized scenarios within the simulation before setting up the required scenarios on the actual virtual machines, thus overall reducing iteration time.

Actual Data Model

The data retrieved from Kali comes in 2 sections, the first section simply outlines the IP address available for scan on the network. The second response outlines the data scanned from each IP. Although, in theory, a scan taking more time, could send the second block of data in an array from multiple different IP, that was out of scope of the first half of this project. In the second half, the research team will looking into scanning ports for all data, and visualize that

information via color coding, and allow for more advanced search functionalities in order to filter and find data pertaining to a specific query for a specific service, with specific vulnerabilities.

```
//******** Network scan of IPs connected the Attacking server **********/
{
   "data": [
       "x1.x2.x3.x4",
       "...",
       "x1n.x2n.x3n.x4n"
   ]
}
```

In the second part of the data model details the ports (web services) along with the application and version associated. This data utilizes the NMap tool, which an example of output from the program can be found on the nmap.org site[6]. The NMap tool enables a few basic network operations, such as scanning the network and the services on that device. Commonly used services that are out of date, are the most likely targets for hackers.

The section for NMap data is under the **scan_target_data** string.To avoid extra time implementing a parsing method for this, the xml string was simply sent along with the rest of the message, and then deserialized on the client side. These services are often patched as soon as the programmer is made aware, however, not all servers apply these patches on time, or the support for that service may no longer be available. With this knowledge, a hacker can look up tools using a program called Searchsploit. Searchsploit is an open source database that details vulnerabilities commonly found in software and the tools that can be used to exploit them. For example, an unencrypted password through a commonly used web service tool can be captured using Wireshark.

In the second part of the message is the **scan_target_exploits,** using Searchsploit, the program then finds a matching exploit to use. Currently the EZhack tool only supported the *unreal_ircd_3281_backdoor* exploit, thus, that value was left as is, since the exploit would only work with Metasploit 2 anyway.

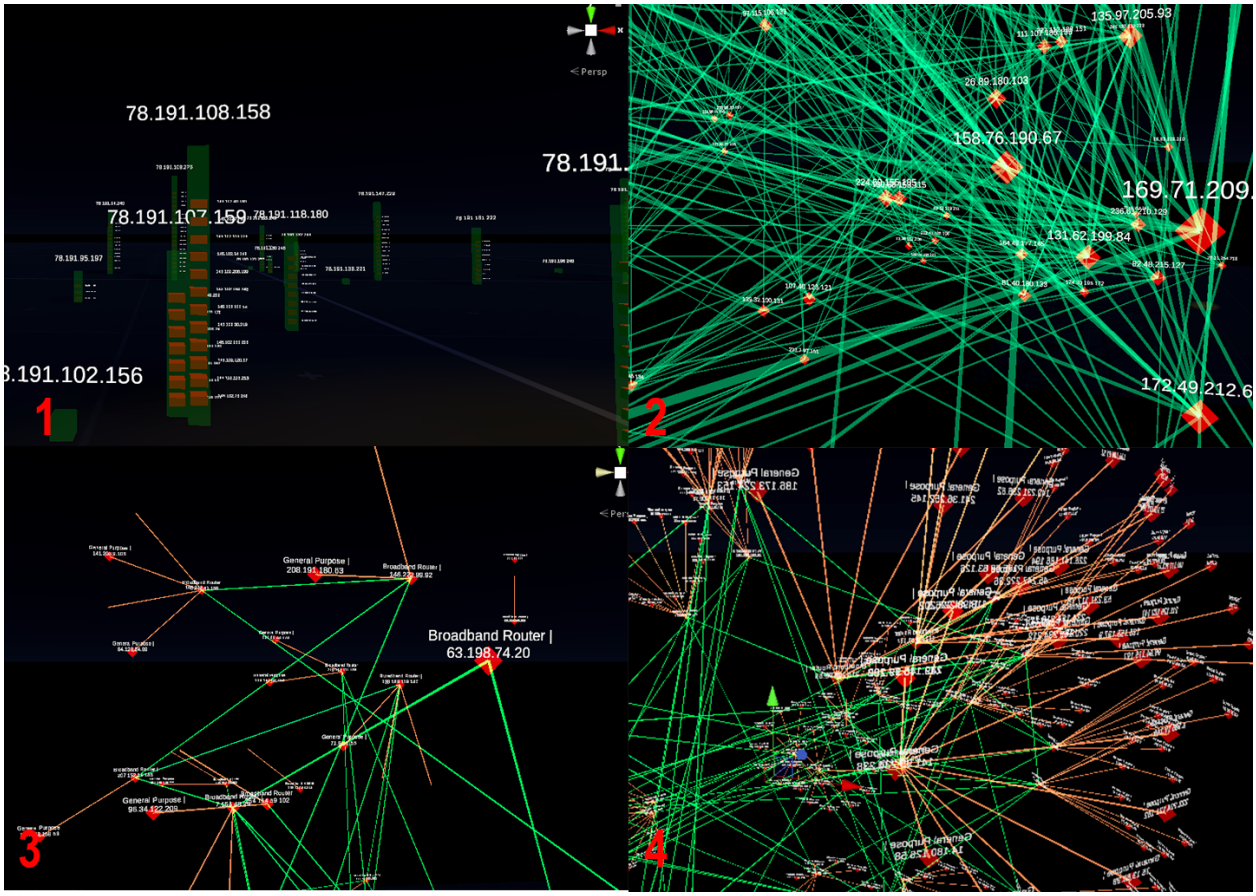
Finally, the last portion of the message **scannedExploits** details the path of the script on the Kali server that contains the exploit script, and allows the program to use the exploit found in a single command. Although, the execution method isn't yet fully implemented on the application, this feature is one of the first features to be implemented in the next phase.The following XML used NMap tools[7] for parsing.

```
//********** The actual payload ***********//
{{
   "message": "target scanned x.x.x.x",
   "scan_target_data": "<XML>NMap Port Data</XML>",
   "scan_target_exploits": {
     "exploit_hardcoded": [
       "unreal_ircd_3281_backdoor"
     ],
     "scannedExploits": [
```

```
  {
    "Author": "Metasploit",
    "Date": "2011-07-05",
    "EDB-ID": "17491",
    "Path": "/usr/share/exploitdb/exploits/unix/remote/17491.rb",
    "Platform": "unix",
    "Title": "vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)",
    "Type": "remote"
  }
] }}
```

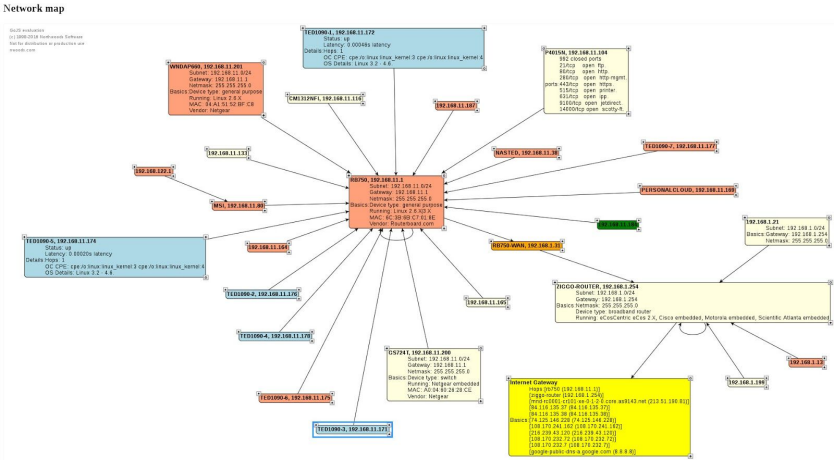

Visualization

For visualization of the data, previous examples (from the previous) work section was looked at in order to find the optimal solution. The way data is arranged is crucial to understanding how it functions in a hierarchy, thus allowing one to navigate through the data and quickly to identify where the vulnerabilities lie. This VR application was implemented using Unity 3D and Steam VR Unity plugin. Through trial and error, and understanding the needs of the user, the design evolved through 4 major stages.



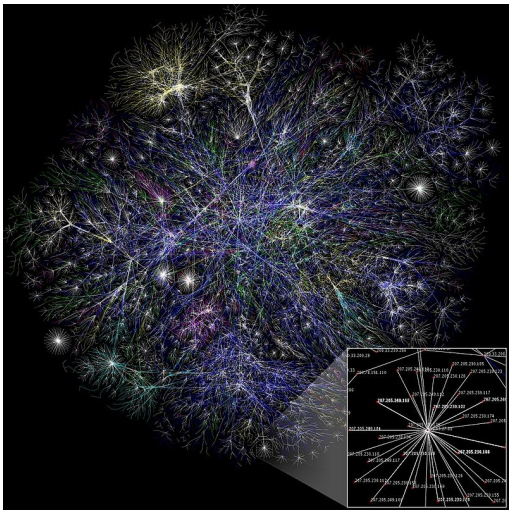
In stage 1, the data was arranged in a blocky format, where there are major networks (in green), and subnetworks in a reddish orange color. The data here would be sparse, but it didn't really represent interconnectedness. Although not a bad design, it was not used for the majority of the first phase of this project, since it couldn't represent the relationships between the nodes in any meaningful way. However, this method is still entirely valid for visualizing data regarding which IP addresses have which ports open. We will revisit this method of rendering data in the future, since its applications could still be very useful, especially with a geographic overlay in place of the 2D plane.

In stage 2, a more connected version was visited, however the data generated made the relationships between the nodes completely random. Also, the data model was still in its early stages, only containing the IP address and the connection it made. Additionally, each IP address referenced other IP addresses in the structure without any clear hierarchy. While looking at an NMap project, we found one particular visualization to be very intuitive [8].



This visual met the requirements of the project, as it allowed the IP addresses to have a central node, the switch (router or modem) connecting it to the outside web. This method of data visualization pointed the team into a direction that would allow remodeling of the data structure in a more robust way.

Stage 3, was when all the references were converted to indices, and types of nodes were also discerned. The data model was still relatively simple at this point, with some minor artifacts in the visualization code, which caused some nodes to not appear correctly since 1 node could not share 2 switches (which would cause diamond inheritance, or network loop). This was closer to the Nmap example, allowing us to embed more data into leaf nodes, which would be the node of interest, and structure routing nodes clustered towards the center, with the servers being at the fringes.

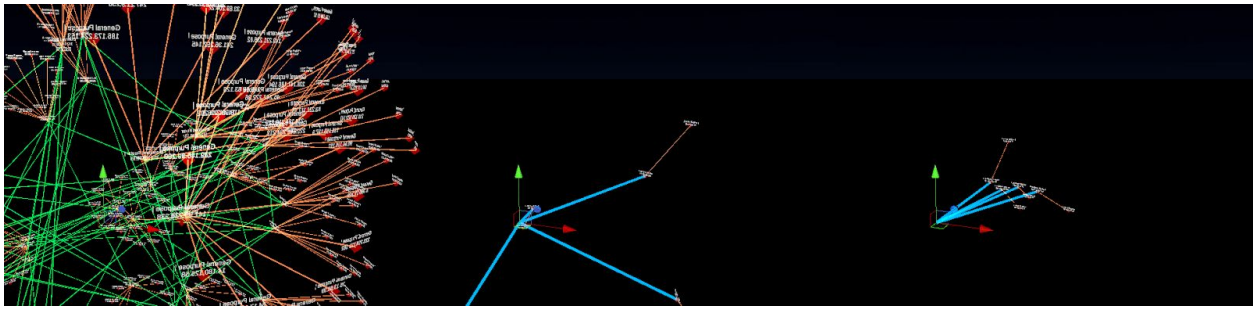


OPTE project Snapshot of the internet from 2005[9]

Stage 4, after looking into various depictions of a network that used a radial design, the team decided to model it as a kind of sphere of interconnected switches, and on each switch was a subnetwork, represented by IP addresses of non-routing devices. These would radiate from a single source, being the point of entry into the rest of the network. This final, and most current stage, displays the routers to be a web of interconnected nodes that are linked by green routes whereas the orange connections went to either a Windows, Linux, or Mac OS device. At this point, the visualizations represented a miniature model of the internet, and thus was the beginning point for the the implementation of functionality for the system.

Animation

After getting the primary look of the network in place, the next step was to look at creating animations that could give the user some sort of intuition as to which region of the network the resulting nodes came from. If we were using a geographic representation, it'd be akin to the user zooming into to a location on Google Maps.



On the far left, the graph is in a "rest" state, which means that the network data (which is simulated data) is unfiltered. When the user punches in a query for a set of IP addresses, the

results are shown in the second frame. The connections on these results switch color to show it has entered a “filtered” state. The results coalesce in front of the user to show the results. This feature is still currently in prototype stage, and will be iterated on heavily. Currently all connections default to the center user, so this will also change to illustrate if these connections are still connected together, or some other hidden switch in the network. More animations, regarding color changes, and also showing the reorganization of switches as data are filtered will also be implemented in the future. Another similar animation feature will also be implemented to the geographic component as well, similar to the Google Maps example stated earlier.

Implementation Part 2:Virtual Interface

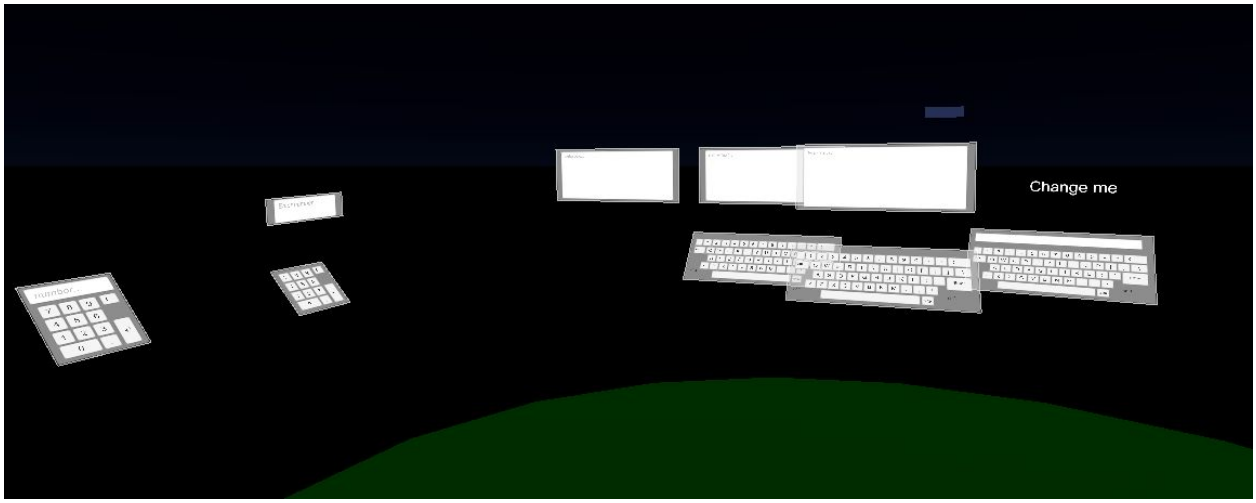
Presentation of the data is one of the major roadblocks the team faced while developing this project. The other roadblock was providing an interface for the user to interact with his or her environment. As the size of the desired network to attack is variable, the team found it important to develop an interface that can dynamically shape the environment.

A necessary goal of development included implementing the interface in a way that can be easily learned and understood by the user. This way, cybersecurity attacks included in this project can be done by any user interacting with the system for the first time, without lengthy training time. In this project, the team investigated integration with Baroque UI as an approach to achieving the above goals.

Integration of Baroque UI

Baroque UI is a VR package that provides user interface elements as a means of visual interaction. These elements are VR objects of typical 2D UI elements that provide a familiar visualization of how to interact with the environment.

In this project, we created a means of summoning a Baroque UI numeric keyboard object that can be used to interact with the data model that is present in the environment. The keyboard can be summoned and removed from the environment at the press of a button. This method of summoning and removal of the UI provides easy access to the user interface when necessary while allowing for the user to clear his or her view when examining the network. Currently, the numeric keyboard is the only UI element present in the program. However, other Baroque UI elements may be investigated for use in other parts of the program. Implementation of the numeric keyboard has given the team a framework for the addition of other elements and provides an initial template for the interaction a user may face.



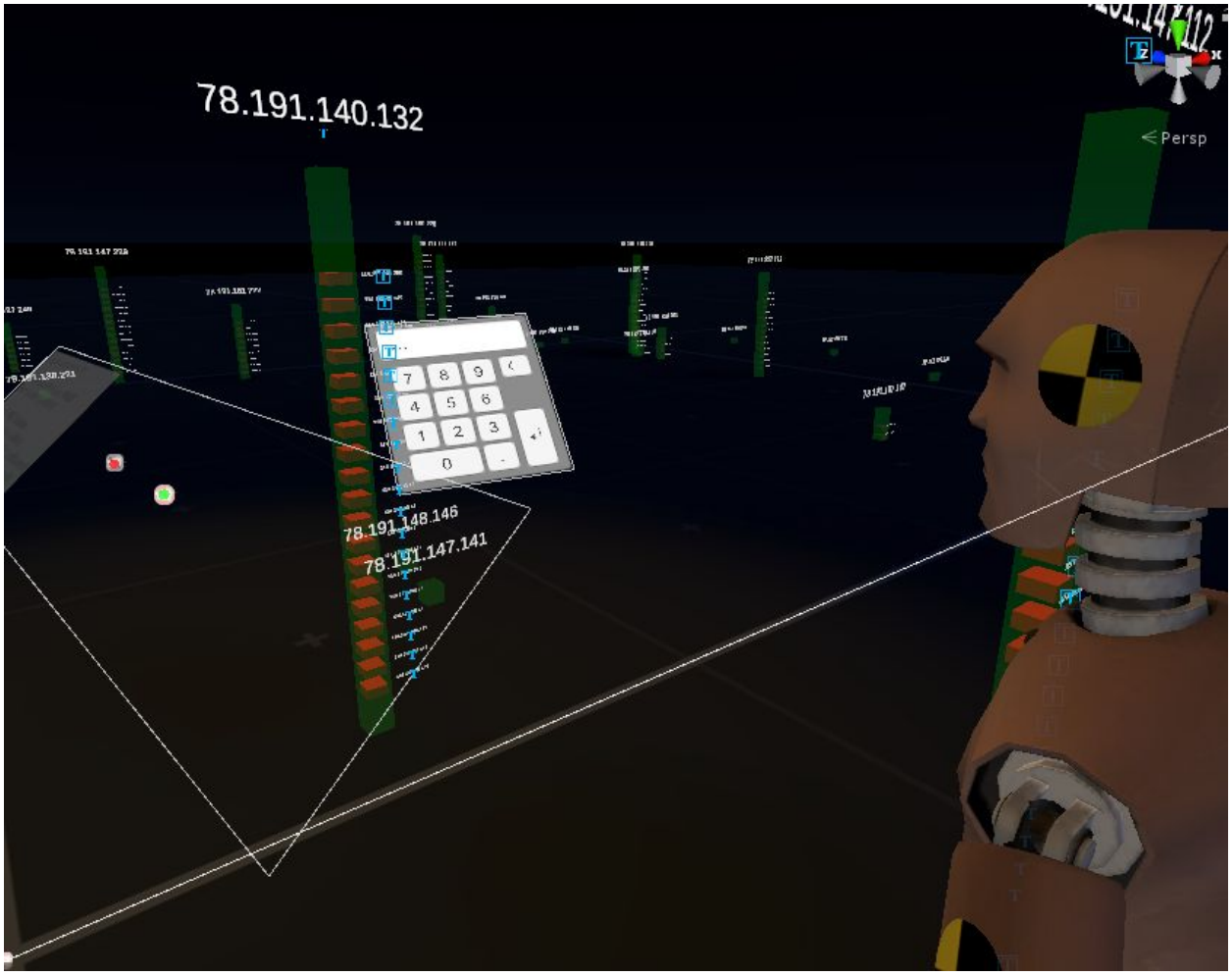
Implementation of the IP Baroque UI

The numeric keyboard that is available for use in the program has been implemented to act as an IP search function. As a user is faced with the sheer amount of networked nodes available for attack, the team found it necessary to develop a method of combing the information quickly and seamlessly for the user. The Baroque UI allowed for the user to input the IP address the user was searching for. As the user entered values into the numeric keyboard, the data model would remove visible nodes from the environment, leaving only nodes that match the input from the user.

Partial IP Search

While implementing the numeric keyboard, the team realized the necessity to provide incremental node removal from view instead of only a direct IP match against the IP the user has inputted. To achieve this, the numeric keyboard provides for partial searching by splitting the input at the decimals of the IP. These partial sections would be compared against each IP address on the network to find suitable matches without the necessity of inputting entire IPs into the interface. Developing this partial search feature has provided faster searching when the user knows the full or a part of the IP address he or she was searching for. However, this method does nothing to provide for searching through the network without already knowing the IP address of the target node. In the future, the team might reapproach this challenge to provide a

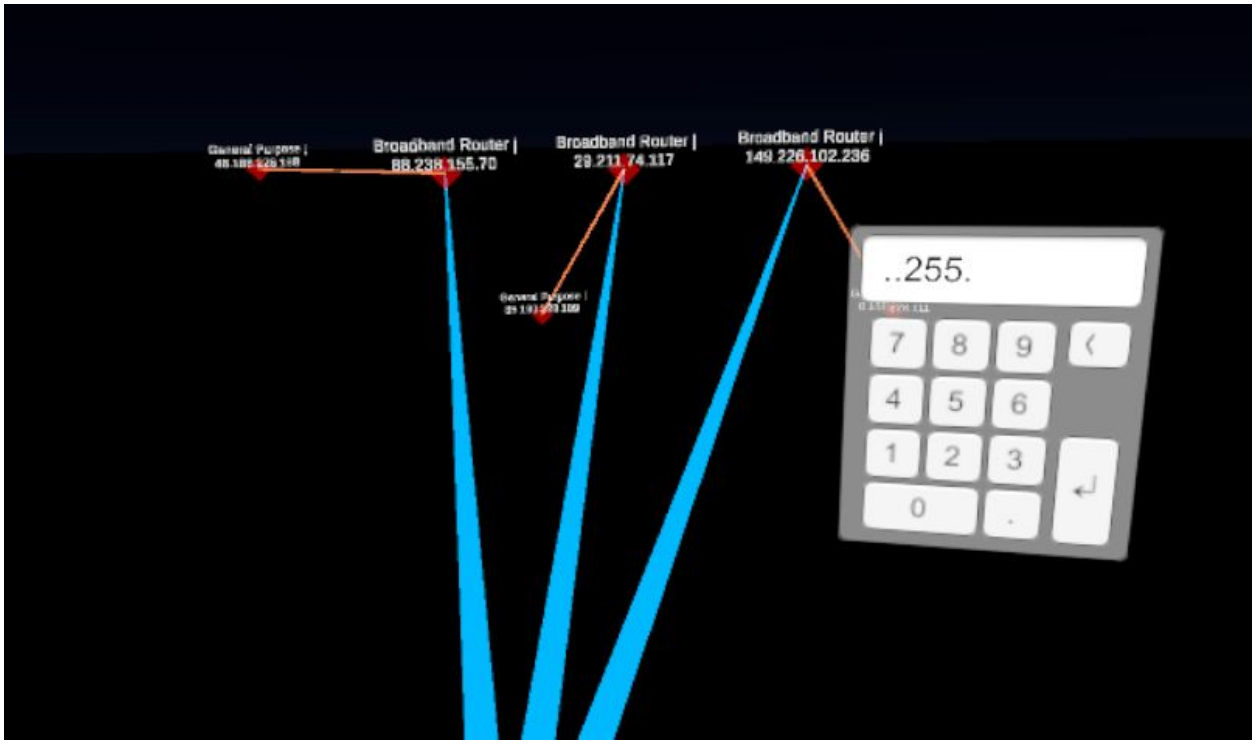
search through the network. A possible solution to pursue would be to search by attributes of a node, such as by possible vulnerability opportunities or recent inactivity.



Alternative Interface Methods

One unfortunate circumstance presented by the Baroque UI the team has implemented is the missed potential of the full application of options available by using the VR environment. The Baroque UI implements 2D elements already in use today on modern computer interfaces. These UI elements were designed accordingly to be interacted with using keyboard and mouse. By using Baroque UI, we are capitalizing on the benefits provided by UI elements that users will already be familiar with using. A typical computer user will already understand how to operate a dialog window or a numeric keypad.

However, by using these 2D elements in the user interface, the program is missing potential in implementing 3D interface methods. There may be potential in replacing 2D elements with 3D elements, whether it may cause improved usability for users not familiar with computer interactions, or possibly by providing improved feedback to the user. Interaction with “real” elements in the world may feel and behave more intuitively in a VR interface and should be considered as a possibility to improve the functionality of VR environments.



Experimentation

This section goes over the actual execution of the code via the Kali server, the extraction of data from that server into the VR application, and the results of the demonstration that the tool can effectively translate network data into a VR environment.

EZHack Integration

The EZhack tool was one of the hardest part to get working. The main problem being minor differences in Python version and Kali server tools version not matching up 1-1 with the original code base. In addition, this tool need a major rewrite, to make it viable for usage as a web service.

First the tool didn't have individual functions to be called while the program ran in a loop, it was nested together as if it were one single program to automate the process in a single host. At the beginning we struggled to get the code to work in its refactored form, and thus each method was compartmentalized, and tested individually using the "Host" object as a central data model.

Through each API (as listed in the design section) a call was made to update the host state. Calling the functions out of order will simply get either a null host, or a default host from the last scanned host. Also, the order of execution for searchsploit work in a similar way, the

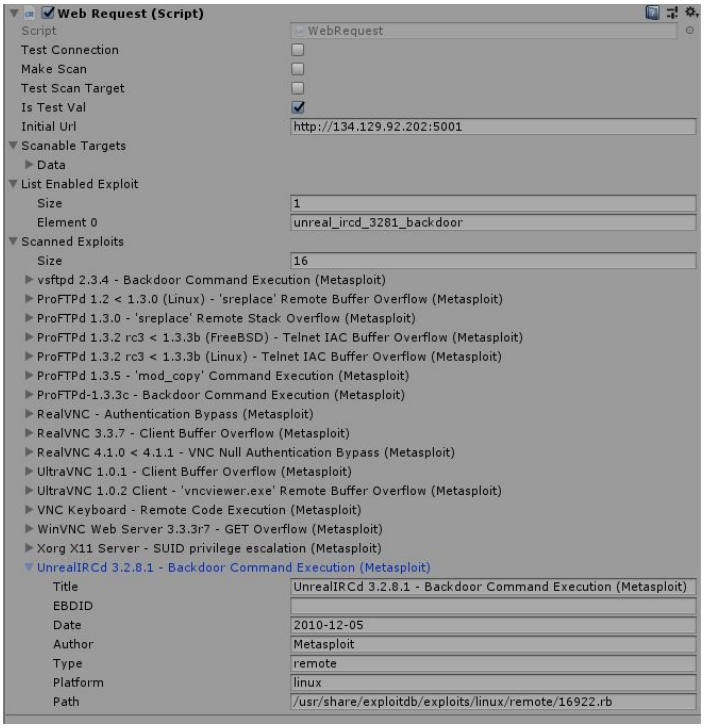
application would have to first get a list of exploit, and then the user would decide which one to execute, and then run a post exploit action (e.g. place a remote execution program on the host). Therefore, the order of operations would rely on the implementation of the client program to call functions in the correct order. Between each function is a decision state that would allow the user to interrupt and branch off into different inputs for each function stage.

Additionally, errors were found in the execution of the Metasploit framework code, where timing was crucial to a functioning set of events to the Metasploit console. This proved troublesome, since the paused time in between each call could be varied. The biggest problem here was that the HTTP calls also had to happen on the server itself via the Python code that opened a connection to a Metasploit console. Eventually, we had to rewrite the remote-procedure-call class, and use a custom timer to get the timing of the functions right in order to read the correct output from the console.

Finally, this output was exported to a json array, and packaged with the NMap data to be sent back to the user. The same console would then later be used to run the execution of the program, and results would be sent back the VR application reporting whether it was successful or not.

Simulation Demo

In this simulation, the EZhack tool uses a hardcoded exploit to make a remote desktop login to a host machine, it then drops a file into a directory that will allow it to execute code remotely by an attacker. Although the visuals are capable of rendering the IP addresses, it was decided to skip that portion for this first phase, since the tool could only return a single array of IP addresses it was connected to. This feature will be added in the second phase of this project.



Without visuals ready, the simulation used built-in Unity editor serialization to display the information in various input boxes. The check boxes represent buttons that trigger various calls. The first checkbox that would initialize IP scan is the Make Scan check box. The second step, Test Scan Target, scans a specified host, in this case a hard coded host in the code, scans the target host for port information, and runs searchsploit to make recommendations for possible exploits to run on the server.

While the execution and exploit selection features work on the backend, they are not yet represented here due to security measures put in place. Since Metasploit is fundamentally vulnerable system, it is not exposed to the external network. Thus, via the Kali server, there is a network switch toggle between external internet, and internal network. The internal allows the server to access the Metasploit server, and execute the actual exploits. Better network solutions may be developed in the future to streamline this process, but may also be simplified to testing as a simulation only mode on the Kali server for the final execution and cleanup phase.

Summary and Future Work

Status	**Goals**
Completed	Create Mockup data to create a sandbox for visual prototyping
Partial	Visual Prototype
Partial	UI Prototype
Partial	UI Integration with Visual System
Completed	API for communication with Kali server environment
Completed	Integration of automated Cyberattack tools
Completed	Test with VR environment
Not Started	Integration with VR environment to generate visuals
Not Started	Integration with UI tools - Command execution from VR environment
Not Started	Implementation of additional tools of the UI for a variety of command executions

The table above outlines the progress of the project. While many of the backend components of the system have been completed, the frontend to allow for user interaction is currently implemented at a minimum level. The goals labeled “Not Started” are intended to be completed in the following semester. Overall, a lot of learning and research had to be done in

the field of cybersecurity itself, as the research team was new to applying their skills in a cybersecurity-oriented setting.

While, the visualization and UI features were more or less based on an architecture that was already thought out, the usage of the EZHack tool seemed to be the biggest surprise. It lacked a lot of features that would make it useful as a server tool, and thus reverse engineering it was a challenge since the team did not have a lot of experience with cyber-security research tools such as these. Overall, the team managed to get the application running, and implemented the primary requirements of remotely executing the automated hacking tools from a VR application.

Future implementation schedule is to get the second phase finished by the end of summer 2019. The next steps are to go even further and to actually carry out an exploit along with planting the file on the host server. The visuals and user interface will also improve to better display, color code, and symbolize the data that has been parsed from the Kali server. The UI will receive an overhaul, contextually representing each phase of the attack. Allowing the user to navigate back and forth between a single host, and view the network overall. By the end of the project, the user should be able to target multiple targets at once, and run the various exploits desired. One of the key goals of this application will be to study how the visuals and UI can work together to allow a non-cyber expert to work quickly in the field using the VR tools and concepts from this project. Once this is done, the team will also look into additional methods of exploiting a host via the VR application Nighthawk.

References

1. Burton, Isaac. *EZHack.* github.com/burtonyaboy/ezhack.
2. Nunnally, Troy, Uluagac, Selcuk, Copeland, John, and Beyah, Raheem. “3DSVAT: A 3D Stereoscopic Vulnerability Assessment Tool for Network Security.” *37th Annual IEEE Conference on Local Computer Networks,* 2012.
3. Best, Daniel M, et al. “7 Key Challenges for Visualization in Cyber Network Defense.” *ACM*, 2014.
4. Hideshima, Yusuke, and Hideki Koike. “STARMINE : A Visualization System for Cyber Attacks.”*ACM*, 2006.
5. Beiró, M G, et al. “A Low Complexity Visualization Tool That Helps to Perform Complex Systems Analysis.” *New Journal of Physics*, vol. 10, no. 12, 2008, p. 125003., doi:10.1088/1367-2630/10/12/125003.
6. nmap.org/book/output-formats-xml-output.html.
7. Vermin, Morbos. “Nmap XML Parsing Libraries.” *NMap Tools*, github.com/MorbosVermin/NmapTools.
8. Sluis, Ted. *Nmap Scan & Visualize Subnets*. github.com/tedsluis/nmap.
9. Lyon, Barrett. *The Opte Project*. LyonLabs LLC, 2005, www.opte.org.