

# Cyber Network Capture Generator

## Project Plan

Sd-May19 Team 5

**Client:** Dr.Benjamin Blakely

**Advisor:** Dr. Thomas Daniels

### **Team Members:**

**Jacob Perin - *Scribe***

**Luke Tang - *Meeting Facilitator***

**Collin McElvain - *Chief Architect***

**Abdelrahman Baz - *Chief Architect***

**Hazem Abdeltawab - *Test Manager***

**Bernard Ang - *Report Manager***

# Table of Contents

<b>1 Introductory Material</b>	<b>4</b>
1.1 Acknowledgement	4
1.2 Problem Statement	4
1.3 Operating Environment	5
1.4 Intended Users and Intended Uses	5
1.5 Assumptions and Limitations	5
1.6 Expected End Product and Other Deliverables	6
2.1 Objective of the Task	7
2.3 Constraints Considerations	9
2.4 Previous Work And Literature	10
2.5 Proposed Design	12
2.6 Assessment of Proposed Design	15
2.7 Technology Consideration	15
2.7 Safety Considerations	16
2.8 Task Approach	16
2.9 Possible Risks And Risk Management	19
2.10 Project Proposed Milestones and Evaluation Criteria	20
2.11 Project Tracking Procedures	21
2.12 Expected Results and Validation	21
2.13 Test Plan	22
<b>3 Project Timeline, Estimated Resources, and Challenges</b>	<b>23</b>
3.1 Project Timeline	23
3.2 Feasibility Assessment	24
3.3 Personnel Effort Requirements	25
3.4 Other Resource Requirements	26
3.5 Financial Requirements	26

3.6 Standards	27
<b>4 Closure Materials</b>	<b>27</b>
4.1 Conclusion	27
4.2 References	27
4.3 Appendices	28

## List of Figures

- Figure 1: Context Diagram for Application
- Figure 2: Data Flow Diagram Level 2 for Application
- Figure 3: Wireframe for Login Page
- Figure 4: Wireframe for Build Page
- Figure 5: Wireframe for Creating a Scenario
- Figure 6: Wireframe for History Page
- Figure 7: Wireframe for Displaying Details of each Test
- Figure 8: Communication Diagram for Xen
- Figure 9: Component Diagram for Application
- Figure 10: Sequence Diagram for Application
- Figure 11: Use Case Diagram for Application
- Figure 12: Gantt Chart for the First semester
- Figure 13: Gantt Chart for the Second semester

## List of Tables

- Table 1 : Major Requirements

## List of Definitions

- ETG : Electronical Technology Group*
- GPL : General Public License*
- RSPAN : Remote Switched Port Analyzer*
- SPAN : Switched Port Analyzer*
- CAPEC : Common Attack Pattern Enumeration and Classification*
- VM : Virtual Machine*

# 1 Introductory Material

## 1.1 Acknowledgement

We would like to acknowledge our client, Benjamin Blakely, and our faculty advisor, Thomas Daniels, for their contributions to our project. Benjamin has dedicated significant time to meet with us every other week as we continue to work out the details of the project. Dr. Daniels has contributed his time and technical advice to our team, as well as assisted in steering us in the right direction on multiple occasions.

## 1.2 Problem Statement

The needs to analyze traffic for hosts, applications, or services is essential in the world of computer security. Traffic is a way of describing how a computer sends information to the internet, and how the computer receives that information back. Traffic analysis is used to detect any malicious or harmful programs that can enter and harms one's computer, like a virus. Thus, preventing any undesired outcomes.

The solution of the problem is to create a program that automatically analyzes traffic data of many types, helping researchers create more innovative ways to combat malwares, and other unsafe softwares. This proposed program will not only serve as a catalyst for researchers to come up with potential solutions, but also provide a simple understanding of Traffic and its effect in computers.

There is also need for the development of a software that analyzes computer traffic for the following reasons:

1. Computer traffic is essential in gaining information that directly affects the inner-structure of a malware.
2. Computer traffic help in creating tests that detect the presence of malware.
3. Computer traffic maintain a safe software in which the hardware runs on.

Thus, our project creates a software that benefit the community in many ways, and not just 1 or 2 ways. We aim for our project to be controlled by Argonne National Laboratory, and they reserve the right to use it either privately or publicly.

## **1.3 Operating Environment**

- For this project, the operating environment will be a web application.
- A web application is convenient because it provides the user with an interface that is admirable and easy to use
- The web application will also be designed to withhold a large number of requests without crashing while the user is typing commands.

## **1.4 Intended Users and Intended Uses**

- Intended Users: our tool will be used by cyber researchers to analyze captures of traffic from a particular combination of host, application, and service. They will be able to use a web application that can automate the traffic generation and analyze it for the scenario they specify.
- Intended Uses: our product will allow a user to select OS, service, and traffic types for a set of servers and clients, and then generate that traffic and analyze it in an entirely automated manner. The above scenario might be for the purposes of traffic engineering, generating training datasets for machine learning, or general host/protocol analysis

## 1.5 Assumptions and Limitations

Assumptions:

- The tool will use a predefined list for traffic types
- The generated traffic file won't be stored locally
- Chef and Xen can be used together to create a properly configured VMs
- Free version of Chef is enough for our purposes
- Being able to run lots of VM without running into problems (having enough memory)

Limitations:

- Open Source Projects only
- Python language use in back end
- Full PCAP storage -- memory constraint
- The tool will be used for research purposes only

## 1.6 Expected End Product and Other Deliverables

- Tool -- Packet Capture Generator
  - Web Application (Front End) End Product
    - Secure/Standard Framework
      - Use of modern web framework -- Django
      - Referenced security best practices for web application
    - Abstracted instantiation of client,server, and attack scenario (pre-define the criteria available)
      - Generate configuration file for client and server setup
        - Specify the OS, Application, Services, etc.
        - Ex:
          - Client: Fedora 9, Firefox
          - Server: Apache, Drupal
          - Network: Proxy
      - Generate configuration file for attack scenario
        - Series of tasks to be executed between client and server

*(This is not fully understood)*

- Ability to view generated output from defined process above
  - PCAP Storage
    - Compressed
    - Full/Partial PCAP Capture
    - Delete/Download PCAP
  - SSH information for created host systems
- Ability to view server health
  - Remaining storage
  - Error logs
- Virtual network environment (Back End) end product
  - Configuration file parsing and execution logic
  - Automated creation of server and client box
  - Automated execution of created attack scenario
  - PCAP Storage
  - Configuration Option Storage
  - Remotely accessible
- Wiki & Source Code
  - Backward tracking of bugs and team member task allocation
    - Fallback Versions
  - Documented design process
    - Failed designs with reasons
    - Logical process leading to final system designs
- Written report:

A report describing the architecture, testing methodology, and expected performance characteristics of the code.



## 2 Proposed Approach and Statement of Work

### 2.1 Objective of the Task

The goal of this project is to create a tool that will help in cyber-related research when there is a need to analyze captures of traffic from a particular combination of host, application, and service. As described in the end product section above, the tool will make it easier and less time-consuming for the researchers to analyze the traffic of interest.

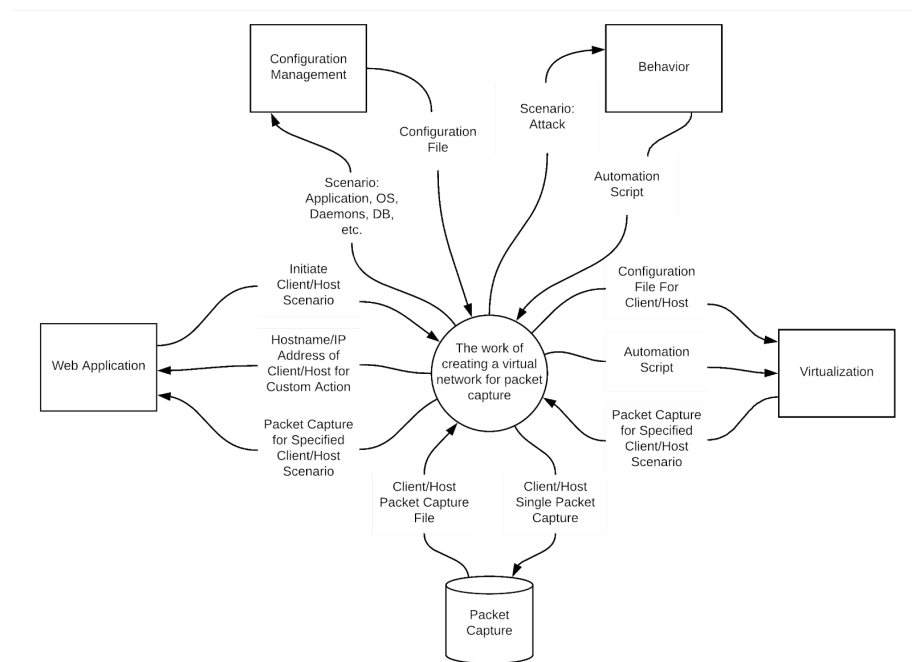


Figure 1: Context Diagram (1st Iteration) for Application

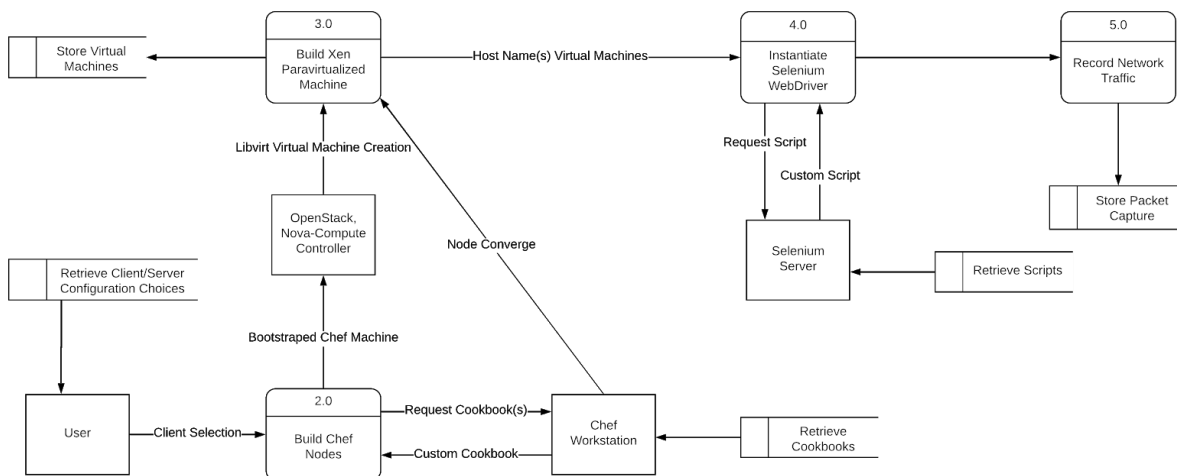


Figure 2: Data Flow Diagram Level 2 for Application

## 2.2 Functional Requirements

### 1. Ubiquitous Requirements

- 1.1. The hypervisor software shall be remotely accessible through a web application (Figure 1: (IN) “Web Application,” adjacent system)
- 1.2. The web application shall provide secure user authentication prior to access (Figure 1: “Web Application,” adjacent system)
- 1.3. The web application shall allow the user to create network capture from pre-determined combination of client, server, daemon(s), application, and activity (Figure 2: User Data Store)
- 1.4. The generated network flow shall be stored in a database for later access (Figure 2: Process 5 Output)

### 2. Event-driven Requirements

- 2.1. When the user selects client/server combination the hypervisor shall allocate and create two separate virtual machines (Figure 2: Process 2 Abstraction)
- 2.2. When the hypervisor has created a virtual machine the configuration management shall establish a connection and load configuration file to virtual machine (Figure 2: Process 2 Incoming Arrows)

- 2.3. When configuration management has initialized a virtual machine the application shall load/execute behavioral scripts on the virtual machine (Figure 1: (IN) “behavior” adjacent system and (OUT) “Virtualization,” adjacent system)
- 3. State-driven Requirements**
- 3.1. While the virtual machines are active the server shall store network traffic to database (Figure 1: (OUT) “capture,” adjacent system)
- 4. Unwanted Behaviour Requirements**
- 4.1. If hypervisor detects insufficient resources then the web application shall restrict virtual machine creation and display “insufficient resources” warning
- 4.2. If server detects insufficient storage then the web application shall display “insufficient storage” warning
- 5. Complex Requirements**
- 5.1. When the network traffic is being stored, if the server detects insufficient storage, then the web application shall display “insufficient storage warning”

## 2.3 Constraints Considerations

### Non-functional requirements

- Performance
  - Demonstration of a working system.
- Scalability
  - Prototype will handle at least 5 virtual machines on a network.
  - Store scenario data for at least 10 one day long worth of traffic flow of the virtualization network.
- Availability
  - Available only to our team of developers and permissioned users during our prototype development.
- Reliability
  - Always properly store compressed PCAP in a reliable manner.
  - Spun up virtual machine scenarios should have a 99% success rate.
- Recoverability
  - No backup data will be required for the prototype development.
- Maintainability

- Be able to continue development of features and bug fixes of the project through the Spring 2019 semester 492 class at Iowa State University.
- Regulatory
  - The majority of software should be written in Python 3.
  - All software incorporated in our project has been selected because of their licensing and open source status. The GNU General Public License (GNU GPL or GPL) is a widely used free software license, which guarantees end users the freedom to run, study, share and modify the software. The Apache License is a permissive open source software license — so users can release modified versions of the Apache licensed product under any license of their choice. Users can freely use, modify, distribute and sell a software licensed under the Apache License without worrying about the use of software: personal, internal or commercial.
    - Xen
      - GNU General Public License, version 2
    - Selenium
      - Apache 2.0 License
    - Chef
      - Apache 2.0 License
      - <https://www.chef.io/terms-of-service/>
- Usability
  - All use case functionality will be accessible through a web application.
- Interoperability
  - Accessible through the Iowa State network for the development team.
  - Virtual networks between the virtual machines should be manageable.
- Cost
  - No costs associated with software as everything is open source.
  - Hardware initial cost and maintainability for hosting VMs, data, server, client information.
- Platform compatibility
  - Web application compatible with any machine capable of hosting any popular web browser.

- Security
  - Any password information will be salted and hashed passwords stored separately from the Server.
  - Any execution of potentially malicious software should be isolated to the virtual network, this will be done with a gateway/proxy to ensure network connectivity to ensure traffic will not leave the environment. In addition any rules for Xen itself may restrict access to the outside network.
- Safety
  - All hardware should be stored and operated in a safe and responsible manner
- Ethicality
  - All work should be original for our development team with credit given to proper sources.
  - No unauthorized copying of software.

## 2.4 Previous Work And Literature

Relevant Team Experience:

- Development experience with programming languages: Python 2.7, 3.7, JavaScript.
- Experience with building and maintaining database systems.
- Experience with lightweight framework and server deployment.
- Experience with web development creating and updating custom webpages.
- Experience with Selenium:
  - Fully functioning web scraper for custom web pages capable of executing javascript programmatically.
  - Web crawler with specific settable parameters to control pace, randomization, and execution time.
  - URL redirects/Web testing using selenium to ensure links properly work and javascript executes correctly under varying controlled circumstances.
- VMware/VirtualBox virtual machine experience for testing with different operating systems as well as examining and testing malware in a controlled environment.
- Projects utilizing these skills can be found here:
  - <https://github.com/Ltango>

- Slack-integrations uses certified server for requests to Slack, uses Python extensively.
  - VTJM uses selenium for URL resolution, uses Python extensively for automation.
- [https://git.ece.iastate.edu/sd/sdmay19-05/tree/Front\\_End](https://git.ece.iastate.edu/sd/sdmay19-05/tree/Front_End)
  - Some of the interfaces we tested and made for the front end

#### Existing market products

- VMware Workstation
  - Vmnet-sniffer: captures all virtual machine traffic to the virtual machines at once to be sorted out later.
    - Isolate HTTP traffic sent through host level load balancer to a random virtual machine.
    - Determine where specific DNS queries are getting picked up.
    - Low level network management packet realizations separate from main network.
  - Works well with Vagrant and CloudShark API requests to automate functions.

#### Relevant market products

- Wireshark and PerformanceVision Virtual Capture
  - Run concurrently in promiscuous mode.
  - Visualize traffic within virtual server.
  - Virtual switch acts like hub for VMs.
- Phantom TAP and GigaVue
  - Intrusive kernel level sends data to an off site analysis device.
  - Sends copies of data from source separate from the network.
  - Is a paid service.
  - May require license upgrade costs.
- Vswitch for VMWare and Openvswitch for Virtualbox/Xen
  - Clean and easy install process.
  - Flexible for multiple purposes.
  - Is a paid service.
  - Requires a ESX license.

- Has a physical switch for RSPAN and ERSPAN capabilities.

#### Difference from market products

- Our solution is completely free.
- Our solution is based entirely on open source material.
- Our solution also comes with an application that will combine resources to automate traffic capture.
- Our solution will generate virtual traffic automatically with prewritten automation scripts.
- Our solution is completely controlled under our own domain without relying on a third party service for data capture or analysis.

#### Citations:

CloudShark. "Packet Capture on VMware Virtual Machines Using Vmnet-Sniffer." *CloudShark Blog*, 9 July 2018, [enterprise.cloudshark.org/blog/packet-capture-in-vmware-virtual-machine/](https://enterprise.cloudshark.org/blog/packet-capture-in-vmware-virtual-machine/).

Rogier, Boris. "Virtual Data Center Traffic Capture and Troubleshooting." *Accedian*, Accedian, 18 July 2016, [accedian.com/enterprises/blog/traffic-capture-virtual-data-center/](https://accedian.com/enterprises/blog/traffic-capture-virtual-data-center/).

## 2.5 Proposed Design

### Front-end:

For the front-end development, we are using Django framework to construct the web application that holds our program together.

Front-end will include the following features:

- Login Page

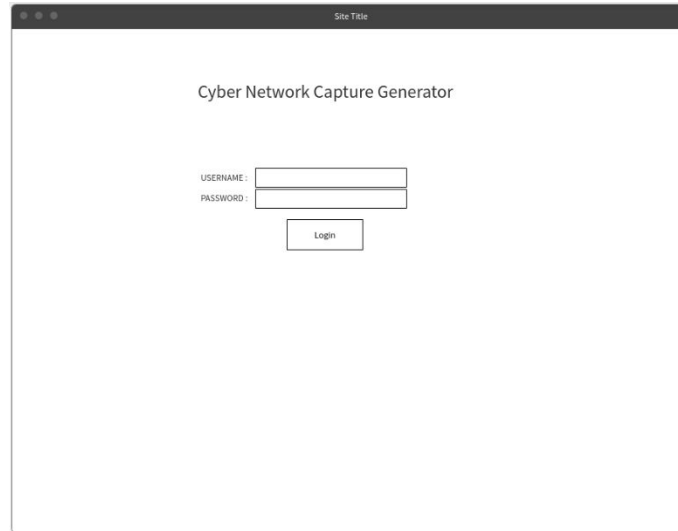


Figure 3 : Wireframe for Login Page

In this template, the user is able to login to the application with his username and password as authenticated.

- Build Page

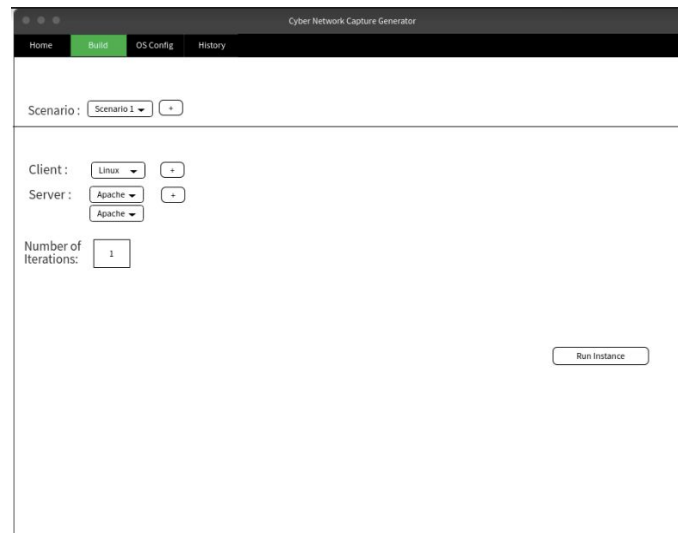


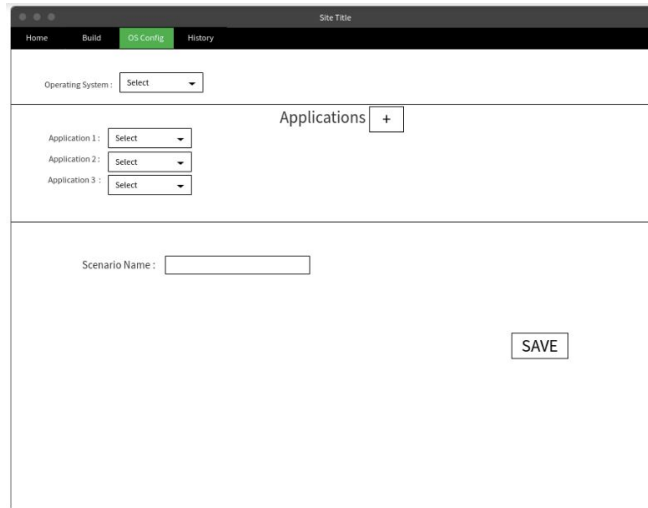
Figure 4: Wireframe for the Build Page

This page can be accessed by clicking build on the toolbar. In this template, the user is able to build a new test by inputting a premade scenario, a client, a server and the number of iterations he/she wants to run them. Note that multiple clients and server can



be added accordingly using the + button. New scenarios can be created in the OSConfig page.

- Creating a Scenario

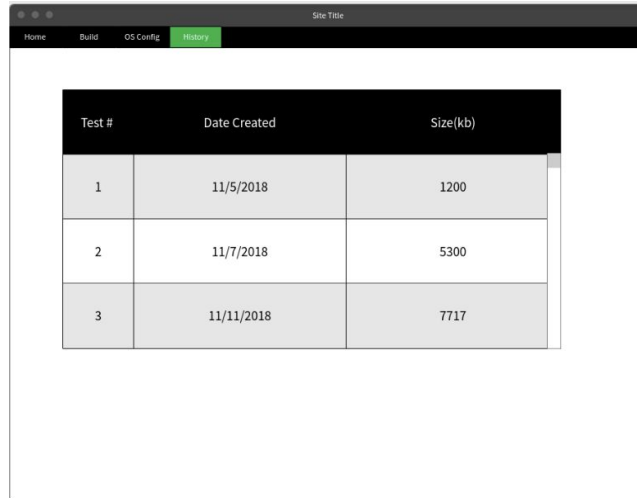


The wireframe shows a web application interface for creating a scenario. At the top is a dark navigation bar with tabs: 'Home', 'Build', 'OS Config' (highlighted in green), and 'History'. The main content area is divided into sections. The first section has a label 'Operating System:' followed by a dropdown menu with 'Select' as the placeholder. The second section is titled 'Applications' and contains three stacked dropdown menus labeled 'Application 1:', 'Application 2:', and 'Application 3:', each with 'Select' as the placeholder. To the right of these dropdowns is a '+' button. The third section has a label 'Scenario Name:' followed by a text input field. In the bottom right corner of the main area is a 'SAVE' button.

*Figure 5 : Wireframe for Creating a scenario*

This page can be accessed by clicking OSConfig on the toolbar. In this template, the user will be able to create a custom scenario and save it. The user can choose the operating system that the scenario will be ran on as well as the applications that the user would want to run in the scenario. Note that multiple applications can be added via the + button.

- History



The wireframe shows a web browser window with a navigation bar containing 'Home', 'Build', 'OS Config', and 'History' (highlighted in green). Below the navigation bar is a table with three columns: 'Test #', 'Date Created', and 'Size(kb)'. The table contains three rows of data.

Test #	Date Created	Size(kb)
1	11/5/2018	1200
2	11/7/2018	5300
3	11/11/2018	7717

Figure 6 : Wireframe for History Page

This page can be accessed by clicking History on the toolbar. In this template, the user will be able to view the previous tests that has been ran by the user. This page shows the date that the test is created, as well as the size of the PCAP file. The user can access the details of the test by clicking on the test number.

- Details of each Test

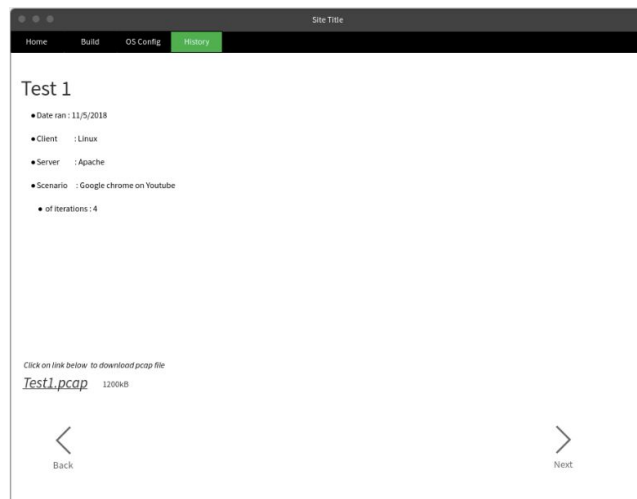


Figure 7: Wireframe for Displaying Details of each test

This page is accessed by clicking on the number of test in the History page. In this template, the user will be able to look at the details of particular tests as well as download the pcap file for the test. As seen, the details include the date created, the client and

server used, the scenario used, and the number of iterations ran. The size of the pcap file is displayed beside the link to download the file. Clicking the next or back button will bring them to the next or previous test accordingly.

## Back-end:

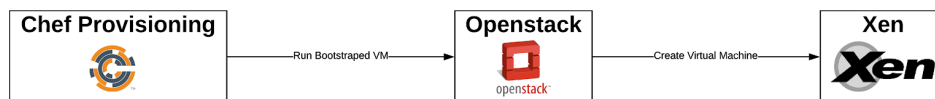


Figure 8: Automated Creation of Virtual Machines

The above diagram represents the creation of the virtual machines that will represent the client and server relationship for a scenario with predefined inputs (defined in the front end.) However, inputs do not magically turn into a configured machine. In order to generate fully functional virtual machines we make use of several open source libraries.

Chef Provisioning allows for “cookbooks” (configuration files) for necessary virtual machine applications, operating system initialization, etc. to be generated on the fly. This is important because each scenario may have a unique set of inputs.

OpenStack is a set of services, however, we only need one part, Nova Compute. This is a controller that will allocate necessary resources and make “hypercalls” (system calls) to the hypervisor.

Xen is a hypervisor. It is capable of hosting multiple operating systems on top of the host using paravirtualization techniques (and others.) This is helpful because it allows Xen to be hosted on most hardware. In addition, it has a defined networking structures for each of the virtual machines. This opens up possibilities for network traffic logging and routing that will be necessary for our project.

After our hypervisor is set up and populated we will want to introduce some traffic to be monitored. There are two vectors of stimulation here; randomized and automated human

browsing, and intentional detonation of nefarious software. The former will rely on Python scripts automating general human traffic with the use of Selenium for its capabilities in executing JavaScript where other libraries such as lib2 are incapable. There is also the added benefit of launching the Selenium WebDriver using different browser binaries e.g. Firefox or Chrome. This will allow multiple randomized and natural instances of web traffic to be generated so that PCAP data can be stimulated and gathered.

Beyond that will be detonation of potentially nefarious software. Because there are so many kinds of attacks all with their own associated risks careful preparations must be made. Even with these preparations there is always a risk of infected files reaching beyond the hypervisor, as malware exists that specifically targets hypervisors, or relaxed rules on features such as shared folders can compromise the entire system. Because of this it is highly recommended to isolate testing machines as much as possible as well as have in depth knowledge of the malware that is being tested.

Now we will be implementing several preventative strategies:

- Isolated network segment will be obtained through the use of firewall rules
- Gateway for the dirty network must route traffic through a proxy, we will be using OpenWRT
- Snort to see alerts will trigger for the host capturing network traffic, inline between a potentially infected host and the gateway.

Following these strategies we will need to introduce the malware files into the virtual environment, execute them, and capture PCAP data for analysis. There will be two methods of introducing malware into the virtual environment: manually, and automatically. Upon setup and configuration, we will set parameters for exactly what we will want running on the virtual machines. Options will include our python human emulation script with parameters on how long to run, how often, and under what web driver binaries (Firefox/Chrome). They will also include a presorted list of basic malware options, what file to introduce, and when to detonate said file. These malware files will need to be stored securely without executable permissions until they have been introduced into the virtual environment. Should other malware options want to be explored, they will have to be manually loaded into the virtual environment and executed.

## 2.6 Assessment of Proposed Design

### Strengths:

- Web framework for front-end makes for a more secure, robust web application
  - Organization of views, routes, and static js/html files
  - Modules relating to security concern
- Xen is an old and boastfully secure product. A lot has been done in the way of promoting security, networking tactics, and easy virtual machine management.

### Weaknesses:

- Design is custom. Support will be minimal. Manually reading through product configuration will be required.
- Xen, although secure and well supported is not the most popular choice. Today a lot of companies make use of KVM or VMware. This results in less “guides” or other relevant documentation for proposed design decisions.

### Trade-offs:

- Multiple open source projects require more configuration. However, this will create a more supportable system in the future.
- Django allows for easy use of python in the backend. Since the “Network Capture Environment” is accessible through python this is preferable.

## 2.7 Technology Consideration

Our project will require building a virtual network based on user input. Therefore, we will be utilizing a lot of scripts. Our team considered two options for our scripting language, Python and Ruby.

The project involves a web application. Our team has a lot of web development experience. We originally wanted to use an MVC framework with CSS, JS, and HTML. However, after looking at the functionality for this web application, as well as the time we have to build the application, we decided a proven web development framework would help with time and the complexity. The team quickly decided to go with the Django web framework for the development of the web application. There are also the free tools that our project will be utilizing. The team researched multiple options for these tools.

When considering the scripting languages to use, Dr. Blakely had recommended both Ruby and Python. The team started off in the direction of Ruby; however, only one member of our team had ever worked with Ruby. Whereas a majority of the team had a lot of experience in Python. This brought us to our conclusion. In order to save time by not researching into Ruby, we decided to use Python as our main scripting language for all of our backend functionality.

The project also requires the help of a few tools for the virtualization aspect of the system. Our team researched multiple different Hypervisors to use for our project. The team knew a lot about VMWare and VirtualBox. VMWare was not an option as a license was required. Therefore, our team studied multiple new hypervisors to use. The two that stood out the most were, KVM and Xen. The research our team did showed that Xen had a lot of flexibility with its multiple hooks. KVM seemed to not be as flexible as Xen, but was still a viable option. The scope of our project brought the team to a conclusion that more flexibility of our hypervisor may be useful down the road. Therefore, we will be using Xen as our hypervisor for this project.

Our project also requires the configuration of multiple virtual machines based on user specification. Therefore, we needed to find a configuration management tool to make this process as efficient as possible. Our team had the value of Dr. Blakely, giving us insight on three major configuration management tools. These tools, Chef, Ansible, and Puppet, are all very popular configuration management tools around the world. However, only two of them are free, Chef and Ansible. Our team made a quick decision on these two possibilities, as Chef used daemons and had a massive library for learning the software.

Another technology that we had difficulties with is Vagrant. We wanted to use Vagrant to spin up the virtual machines on our network. However, the connection between Vagrant and Xen is not as easy as we expected. This required the inclusion of the Libvirt library. A library that integrates Vagrant with Xen. Once we found this library we have had much more success with communicating to Xen via Vagrant.

## **2.7 Safety Considerations**

The overwhelming majority of development work here is software development that has no safety considerations. The only safety concerns exist around the transportation, storage, and maintenance of hardware.

## 2.8 Task Approach

1. Configure and launch standalone virtual machine using Xen.
2. Configure Chef to be able to ensure compatibility with the project.
3. Capture PCAP traffic in the virtual machine.
4. Configure and launch specific virtual network with 2 virtual machines.
5. Capture PCAP traffic in the virtual network.
6. Develop a server to handle basic requests and integration with a database.
7. Establish database/preliminary storage.
8. Establish front end web application.
9. Store preliminary PCAP data in database/preliminary storage through server.
10. Develop Vagrant to handle required requests according to Figure 1.
11. Develop Chef to handle config requests according to Figure 4.
12. Develop Database (or preliminary storage):
  - a. Establish tables
  - b. Establish dependencies
  - c. Establish meaningful requests within the scope of our project
13. Develop Server:
  - a. Create and Handle Xen VMs
  - b. Create and Handle Chef
  - c. Handle data to and from database
  - d. Accept and respond to requests from the front end web application
  - e. Safe and secure
  - f. Handle compressed PCAP files
14. Develop Web Application:
  - a. Establish interactivity that suits all use cases as described in figure 5
  - b. Make user friendly and appealing

15. Develop Automation Scripts:
  - a. Selenium to generate proper traffic
16. Develop Analysis tools for PCAP data.
17. Work on stretch goals:
  - a. Multiple operating systems
  - b. Develop users functionality
  - c. Cloud interactivity

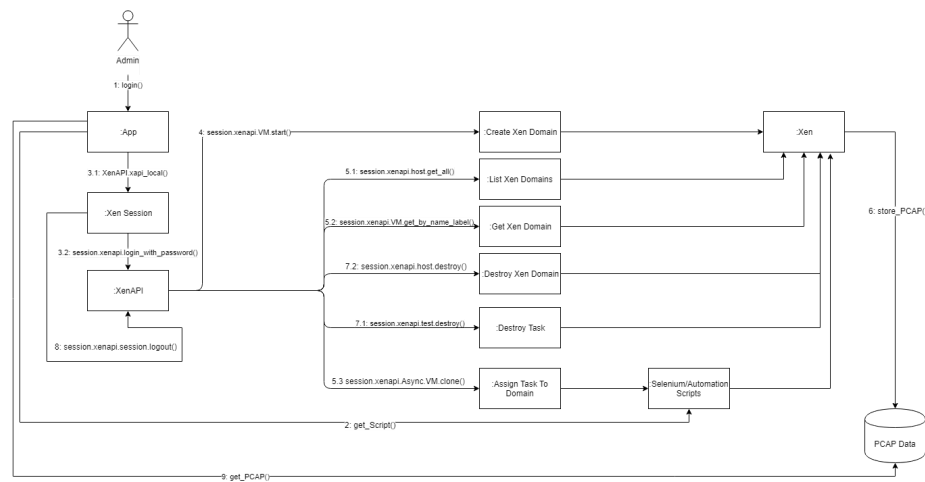


Figure 8: Communication Diagram for Xen

This figure describes how Xen and the Xen API will be interacting with the front end web application and database through specific calls.



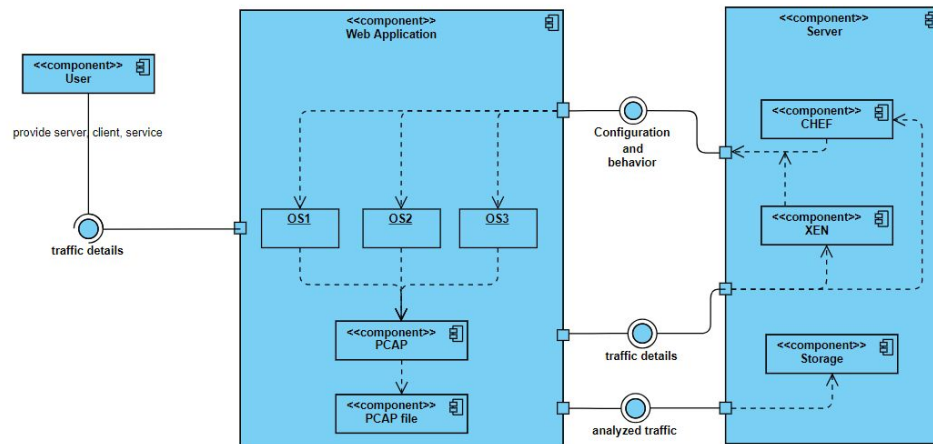


Figure 9: Component Diagram for Application

This figure shows how all components of our system fit together and where they will reside.

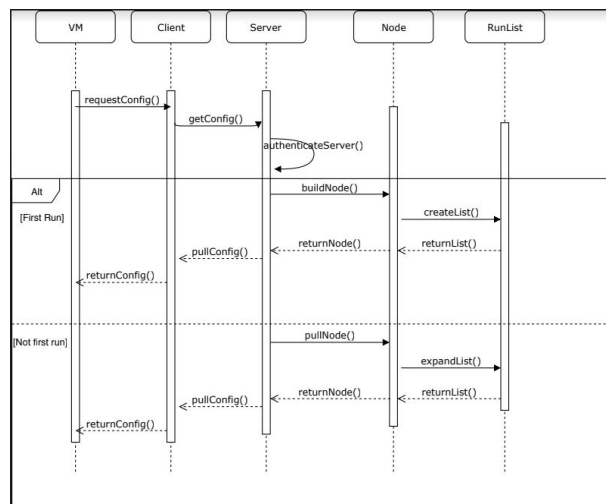


Figure 10: Sequence Diagram for Application

This figure shows the sequence of events between our components for chef's functionality.



Figure 11: Use Case Diagram for Application

This figure shows all front end enabled use cases for the entirety of the project.

## 2.9 Possible Risks And Risk Management

Possible risks

- Incompatibility of software involved:
  - Xen needs to handle applications and scripts launching automatically, failure to do so would be a complete failure of the system.
  - Chef needs to work with Xen to automate configuration, without this step manual scripts or another application will need to be responsible for automating configuration of the virtual network.
  - Front end application will need to handle all use cases and properly communicate with virtual network, server, and server to database. An improper implementation of that interface will make the application useless

- Hardware limitations:
  - Supplied hardware may not be able to hold as many scenario data as we would like. This factor will be unknown until testing stages of the system.
  - The application may not be able to launch complex virtual networks.
- Time constraints may limit functionality of the system.
- Lack of expertise in virtualization of a network may lead to delays in the development process.

#### Risk Management Strategies:

- Communication in development:
  - Slack as main conduit of centralized and recorded communication for the development team.
  - Properly documenting and commenting on developed code.
  - Proper use of GitHub for efficient and and progressive development.
- Sharing expertise and knowledge:
  - Code reviews will be conducted weekly during development sprints.
- Shrinking prototype executables to fit hardware limitations - or obtain more or better hardware to meet our goals.
- Project planning and progress reports ensuring minimization of dead time:
  - Use of Trello for task completion and visualization of progress.

## 2.10 Project Proposed Milestones and Evaluation Criteria

- **Milestone 1:** Install and remotely configure XEN, hypervisor, running on server
  - **Test 1:** Ensure hypervisor properly allocates resources
  - **Test 2:** Ensure hypervisor and server are configured to the network and accessible from other machine.
- **Milestone 2:** Utilize Chef, configuration management tool, to standardize generation of predetermined list of daemon and application
  - **Test 1:** Ensure Chef is configured properly on server to access nodes (created virtual machine)

- **Test 2:** Ensure Chef Cookbooks (config files) are properly accessible by correct nodes
- **Milestone 3:** Generate and configure virtual machines running multiple operating systems, initial setup scripts, and initialize as node on chef network
  - **Test 1:** Ensure initial daemon with chef-client is accessible from chef workstation (located on server.)
  - **Test 2:** Ensure chef node properly pulls updates from server
- **Milestone 4:** Generate behavior of created virtual machine builds, filter to specific traffic, and store to in PCAP and Netflow.
  - **Test 1:** Ensure PCAP is properly stored to database
  - **Test 2:** Ensure PCAP collects correct data
  - **Test 3:** Perform load testing on database when multiple virtual machines are running and storing data to database.
- **Milestone 5:** Build and test the web interface using Django Framework and make sure it works and looks according to the client needs. Also to test the pairing of Django and Apache Webserver
  - **Test 1:** Ensure that the web interface fits its requirements
  - **Test 2:** Ensure that the web interface works dynamically
  - **Test 3:** Perform some tests on sending and receiving data with Django and Apache

## 2.11 Project Tracking Procedures

- Team Work Allocation:
  - <https://trello.com/b/elddC5KG>
  - Pool tasks (To Do) and delegate to proper team members in team meeting
- Weekly Status Reports and Project Documentation:
  - <https://sdmay19-05.sd.ece.iastate.edu/>
  - Testing documentation, Github activity, Team member information, and Posted Status Reports
- Github
  - Documented wiki on design choices and project features

- Guided walkthroughs on development environment and necessary box setup
- Progress in code development
- <https://git.ece.iastate.edu/sd/sdmay19-05>

## 2.12 Expected Results and Validation

Our end goal will be a fully functional testing web application. The web application will allow the user to create different testing scenarios for a virtual network. The user will then be able to test different malware and applications on their created virtual machines on the virtual network. Our team plans to have extensive unit testing on these front-end functionalities. Our team has prioritized the usability of the application. There will be testing on all functionalities of the front-end after any change. These tests will be run through Hazem.

The application will be able to create up to 5 VM's on one network. The user is able to manipulate these machines on the web app through the configuration pages. The user will also be able to configure how the machines talk to each other. This includes the traffic protocols sent (HTTP, IMCP, HTTPS, SMTP, etc.), the amount of traffic sent, and the direction of the traffic throughout the network. The team has scalability as one of our top priorities. The team will run stress tests on the system throughout the designing process, in order to, get to the desired usage goals.

The user will also be able to get the traffic data from their loaded scenario. The web application will allow for filtering and downloading of the desired traffic on the virtual network. The system will save all of this data into PCAP files. The capturing of the traffic is the main purpose for this project. Our team will be testing this traffic filter and capture on multiple different browsers.

We plan to validate the project by going through our final test plan of the application. This should give good insight to how the web application accomplishes all of our use cases as well as showing that functions work as expected. Our group will also run through multiple test iterations with the client to see if there are any minor functional changes the client would like.

## 2.13 Test Plan

This project has multiple components to it. Therefore, the team has planned for thorough testing of the system. Most of the tests will be integration tests as we are using multiple tools to come to our one solution. We will also have front-end testing on the web application that will be primarily based on the usability and user experience.

Throughout the design and development of the system, our team will have a test plan that we will be adding to as problems arise. This test plan will be used when unexpected problems or unexpected values occur. The team will update the test plan to account for these problems in the future testing of the component. The team plans to run through this updated test plan as our final run through of the entire application and system.

The integration testing will begin at the start of development. Our group will begin the development of the client and server. The first few integration tests between the connection of our client and server will begin.

From here we begin the integration of our hypervisor and our configuration management tool, Chef. The connection between Chef and our hypervisor will be crucial for our project to succeed. All of the integration testing for this connection will be run from the individual boxes themselves, as well as from the Chef Workstation. These tests will be looking for edge cases from possible inputs into the configuration management tool.

The integration of the traffic data from the system to the database and then to the web application will account for a lot of the testing time. This data must be able to be captured, saved, and filtered. This is the most important aspect of the system. Our team will be testing every available protocol, as well as filtering through all possible combinations. The team will also be pushing the system to its data capturing limits through stress tests.

The front-end of this project must be user friendly as well as very hands on with the scenarios. Our team will be creating multiple scenarios in order to find any headaches or problems with the front-end functionality. We will be going over the visual appeal of the front-end as well.

The final testing will be done by the entire team over the whole system. This testing will be through the team-contributed test plan that we made throughout the development process. This will include all problems that we ran into, as well as small unit tests.

## 3 Project Timeline, Estimated Resources, and Challenges

### 3.1 Project Timeline

#### Semester 1 → Gantt Chart

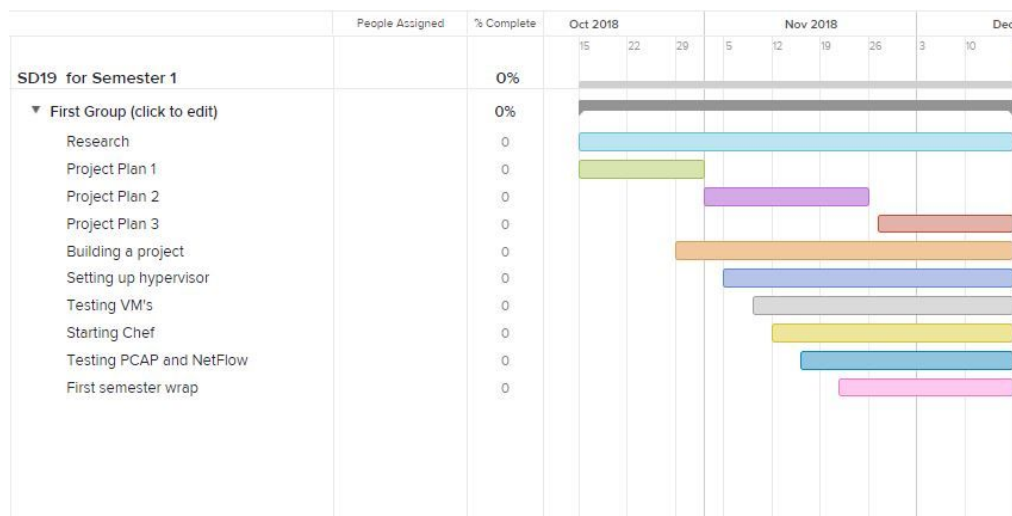


Figure 12 : Gantt Chart for first semester

- The Research above means to keep learning as we go through projects, and not learn a specific task and then try to brute force the project at one.
- The **3** tabs describing the project plan are important because they provide us with a series of steps to help implement our project on time.

- Building a project is building an interface that holds all pieces of our project. For example, a website that holds all aspects of the project together
- Setting up hypervisor is one of the crucial steps in this project. A hypervisor acts as the brain of our program, which tells every component how to do the job.
- Testing VM's is starting a couple of virtual machines and testing them before applying them to our project.
- Starting Chef is analyzing and configuring our Chef script that will eventually be responsible for creating our virtual machines.
- Testing PCAP and Netflow is working with these types of traffic recorders before including them in our project.
- First semester Wrap is when we review everything and make sure every component is working well separately, before merging all of them together in the second semester.

## Semester 2 → Gantt Chart

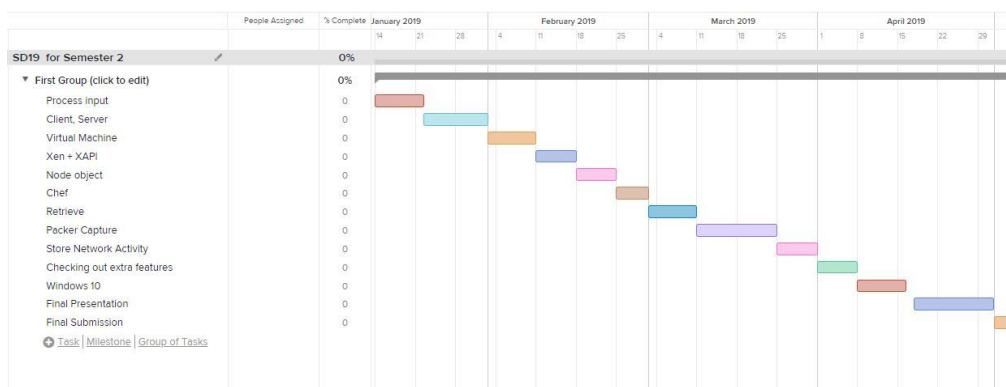


Figure 13 : Gantt Chart for second semester

- Configuring a user input such that, when the user commands something, the program retrieves that command and does some work in the back end.
- Connect the client and server to the hypervisor before adding any other parts to the project..
- Adding the Xen software to the hypervisor to enable direct commands between the two of them.



- Adding Cookbooks and Allowing Chef to send commands to the virtualization software, and creating virtual machines with certain requirements via Xen.
- Configuring Packet Capture (PCAP) and Netflow to record traffic arriving and leaving from one certain virtual machine.
- If time allows, add extra features that the clients would like to be added in the program, such as virtualization for Windows 10.
- Prepare for final presentation and final submission in the last 2-3 weeks of the semester.

## 3.2 Feasibility Assessment

### Risk Assessment:

- End product goal is ambiguous. Although individual parts (initiate hosts, generate behavior on hosts, etc.) are defined, the approach itself is unclear.
  - Ex: The “Attack Scenario” is a series of actions. The extent of this is unknown. Creating the behavior of scenario itself, such as: generating specific web traffic traffic, attack execution, etc. is going to be heavily scenario dependent and possibly unique for each scenario. How to approach this in an automated system is not clear.
- Project has no clear completion criteria. If we are able to initialize a set of hosts for an attack scenario this does not define “completion.” Notion has been made toward several network protocols and attack vectors, however, this has been and will be subject to change as project progresses.
- Project is at risk to not be completed in allotted time frame, for such reasons:
  - Students designing the “network capture generator” have no experience in system design. This has, and will lead to failed design decisions.
  - Python, although ideal for back end glue is not familiar to any of the students. Learning and design patterns will be imperfect, and subject to frequent change.
  - Students designing the front end have little to no experience with web frameworks or best practices. This will lead to slow development, and will most likely need to be changed frequently.

- Scope of work is fluid. The actual extent of “attack vector automation” is unknown.
- Using multiple open source projects such as xen, openstack, and chef is possible. However, this is a lot. Especially for a student who is new to concepts such as “virtualization”. This risk, over time propagates. What seems manageable becomes less manageable a week later when student learns new concepts.
- Scripting a single attack is manageable. However, creating an environment that is built to run different attacks remotely on hosts is difficult. How the server stores attacks, accesses, and executes attacks on virtual hosts will define the manageability of the system itself.

#### **Cost Considerations:**

- Project makes use of open-source projects and is software based. DC

### **3.3 Personnel Effort Requirements**

#### **Major Requirements:**

<b>Task</b>	<b>Description</b>	<b><i>Estimated Time Required</i></b>
Research Xen	Research on how Xen suits our requirements and how to use it for our project	10 hours
Research Chef	Research on the Chef suits our requirements and how to use it for our project	10 hours
Research pairing Xen and Chef	Research on how the pair of Xen and Chef work side by side each other to achieve our requirements	10 hours
Research Django Framework	Research on how to use the Django Framework to develop our front end	10 hours
Research Apache	Research on how to use the Apache webserver	10 hours

Webserver		
Research the pairing of Django and Apache Webserver	Research on how to pair Django and Apache side by side for our interface and server functions	10 hours
Research on other choices of builds and hypervisor pairings	Research on the other choices that are available to us to use as a suitable technology and comparing them with our current choices.	30 hours
Testing Virtual Machines	Come up with several test cases or scenarios that we can run on the virtual machine according to our needs	50 hours
Testing PCAP and NetFlow	Come up with several test cases or captures that can be used as examples of outputs we would want according to our needs	30 hours
Setting up Hardware	This task requires setting up hardware in the department lab to be used as a “server” to run our Virtual Machines	10 hours
Design Project Layout	Developers are required to come up with a design of the interface of the project	20 hours
Setup Xen	Setting up Xen to work in ways that fits our needs and requirements	30 hours
Setup Chef	Setting up Chef to work in ways that fits our needs and requirements	30 hours
Develop Frontend	Developing the frontend of the project, such as the layouts and the webpage using various	60 hours

	libraries .	
Develop Backend	Developing the backend of the project, such as integrating the automation of the virtual machines that will be used in our project.	90 hours
Testing Full Project	This will require the develops to run multiple test cases to test the major functions of our project and testing the final project to work according to our requirements	50 hours
Beautifying layout of Project	This will be a optional task that requires the developers to further beautify the layout of the project and make it more user-friendly	20 hours
Documenting Software	Members are required to use proper documentation of all code, design pattern and architectures used throughout the project	100 hours

*Table 1 : Major Requirements*

### 3.4 Other Resource Requirements

To conduct the project, there are a few other resources that are required on both the software and hardware sides. On the software side, the team will be using GitHub for code control as well Google Drive to maintain the documentation required for the project. The Google Drive storage will be provided by Iowa State University while GitHub will be provided by the Electrical and Computer Engineering Department's Electronics and Technology group. To draw our diagrams, we will be using both MockFlow and DrawIO. We also will be using NotePad++ as our preferred IDE. NotePad++ is provided as a free and code produced on it can be used/redistributed under the terms of the GNU General Public License. As for the development of the frontend, we will be making use of the Django Framework, which is a free and open source web framework that is written in Python. As of now, we are considering the use of either Apache or Nginx for our

webserver. On the hardware side, the team will be using 2 desktops provided by the Electrical and Computer Engineering Department's Electronics and Technology group to be used in the department's lab as our "server" that will be running the virtual machines as well as databases.

## 3.5 Financial Requirements

The project is proposed to be developed cost free. The hypervisor used for the project, Xen, is a free and open-source software that is only subjected to the requirements of the GNU General Public License. For the configuration management of the virtual machines, the team will be using Chef. Chef has a paid as well as a cost-free version. The cost-free version of Chef will most probably be enough to support the requirements needed for the project.

## 3.6 Standards

### PEP8

- Code styling standard for all python code

### CAPEC

- Active catalog of attack vectors
- Tasks executed in our application for any given attack scenario should correlate directly to the matching vector described in cataloged attack scenario

### IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks

- Background: Network developed must contain malicious traffic to specific host and client. Accomplished by bridging the traffic generated by each host through a proxy.
- Section 7 as outlined in this document describes best practices when bridging a network of VLANs. Incorporating this standard will prevent network leaks that could result in consequences for victim users outside network.

# 4 Closure Materials

## 4.1 Conclusion

Cyber attacks are becoming an increasingly bigger problem for the world every day. The problem with trying to prevent any attack on a system is that we can only stop what we know of. Therefore, the best mitigation for stopping any cyber attack on a system is to test the actual attack on your system. You can then identify the symptoms of those attacks on your system when they are attempted.

The only way to safely test a malware on your own system is through a sandbox of your system. This would be a virtual environment where you can record and control the activity of your system and the malware. There have been tools and software made just for these tests; however, these tools and softwares are not cheap and can sometimes be very strict on their configuration.

Therefore, our solution is to create a free, fully configurable testing software. Our team will build a web interface that will control a virtual environment. This will allow users control over the systems added to the environment as well as control over all the data sent over the virtual network. This solution will help push the research for cyber-security forward.

## 4.2 References

Websites used for Research of Tools:

1. <https://www.xenproject.org/>
2. [https://wiki.xenproject.org/wiki/Xen\\_Project\\_Beginners\\_Guide](https://wiki.xenproject.org/wiki/Xen_Project_Beginners_Guide)
3. <http://www-archive.xenproject.org/products/xenhyp.html>
4. <https://www.xenproject.org/developers/teams/xapi.html>
5. [https://docs.chef.io/chef\\_overview.html](https://docs.chef.io/chef_overview.html)
6. <https://www.seleniumhq.org/>
7. <http://www.tcpdump.org/manpages/pcap.3pcap.html>
8. <https://www.solarwinds.com/what-is-netflow>

9. <http://xapi-project.github.io/xen-api/usage.html>
10. [https://wiki.xenproject.org/wiki/Xen\\_Project\\_4.9\\_Man\\_Pages](https://wiki.xenproject.org/wiki/Xen_Project_4.9_Man_Pages)
11. [https://wiki.xenproject.org/wiki/Xen\\_Networking](https://wiki.xenproject.org/wiki/Xen_Networking)
12. <https://help.ubuntu.com/community/Xen>
13. <https://discourse.chef.io/t/resolved-advise-on-first-chef-invocation-on-a-vm/2847/3>
14. <http://www.tomsitpro.com/articles/configuration-management-tools.2-920.html>
15. <http://www.admin-magazine.com/Articles/Automating-with-Expect-Scripts>
16. <https://learn.chef.io/#/>
17. <https://docs.djangoproject.com/en/2.1/howto/deployment/wsgi/modwsgi/>

## 4.3 Appendices

All diagrams were used in explaining the parts