Cyber Network Capture Generator

Project Plan

Sd-May19 Team 5

Client: Argonne National Laboratory

Advisor: Benjamin Blakely

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

List of Figures

List of Tables

List of Definitions

ETG : Electronical Technology Group
GPL : General Public License
RSPAN : Remote Switched Port Analyzer
SPAN : Switched Port Analyzer

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 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 related 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.

1.3 Operating Environment

For this project, the operating environment will be a web application. A web application is popular because it provides the user with an interface that is easy to use and ensures a user-friendly experience. Additionally, The web application will be designed to withhold a large number of requests without crashing while the user is writing 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

Limitations:
- The project is cost free
- PCAP files are very large that we might run out of space

1.6 Expected End Product and Other Deliverables

- Web application:
  Our end product will be an extensible web application tool that will allow a user to select the type of OS, Service, and traffic types (from a predefined list) for a set of servers and clients, and then, in an automated manner, create and configure the necessary virtual machines, initiate a packet capture (saved to PCAP files) and the specified traffic type, and save the capture for later analysis.

- PCAP files:
  Packet capture files of the generated traffic, for some different combinations of OS, service, traffic type, saved for later analysis.

2 Proposed Approach and Statement of Work
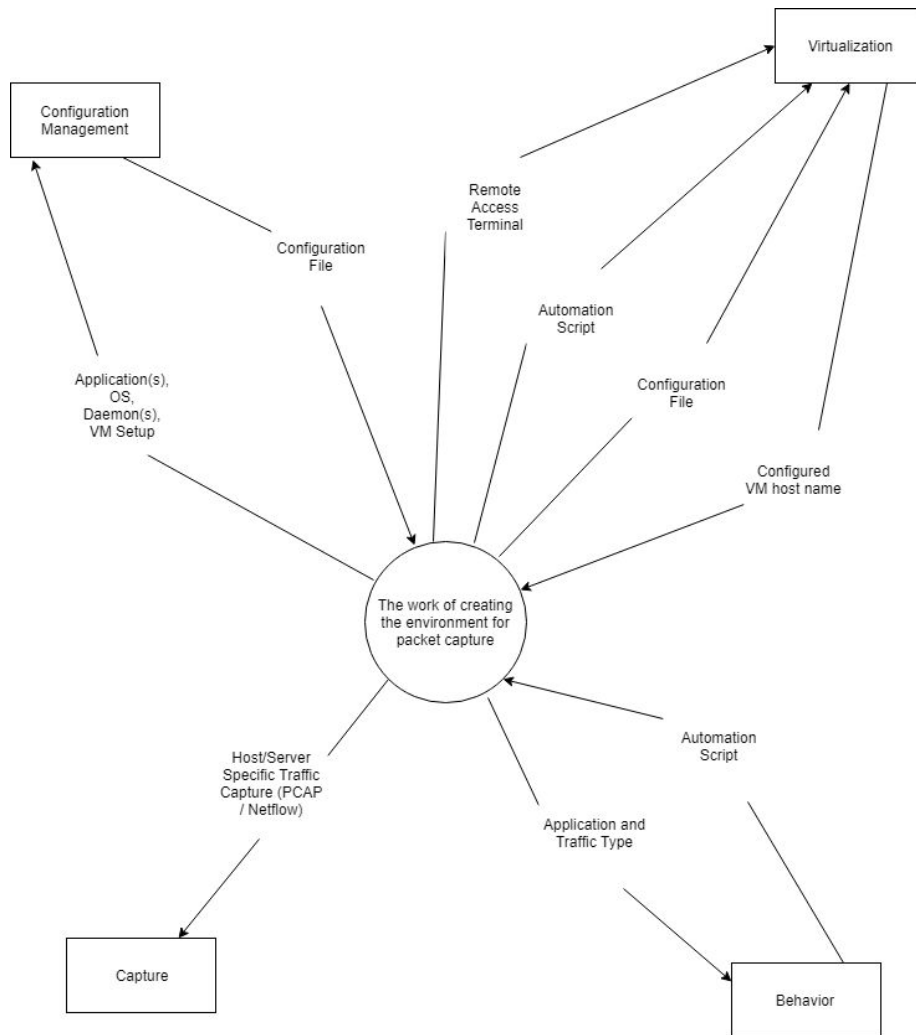
Figure 1: Context Diagram (1st Iteration) for Application
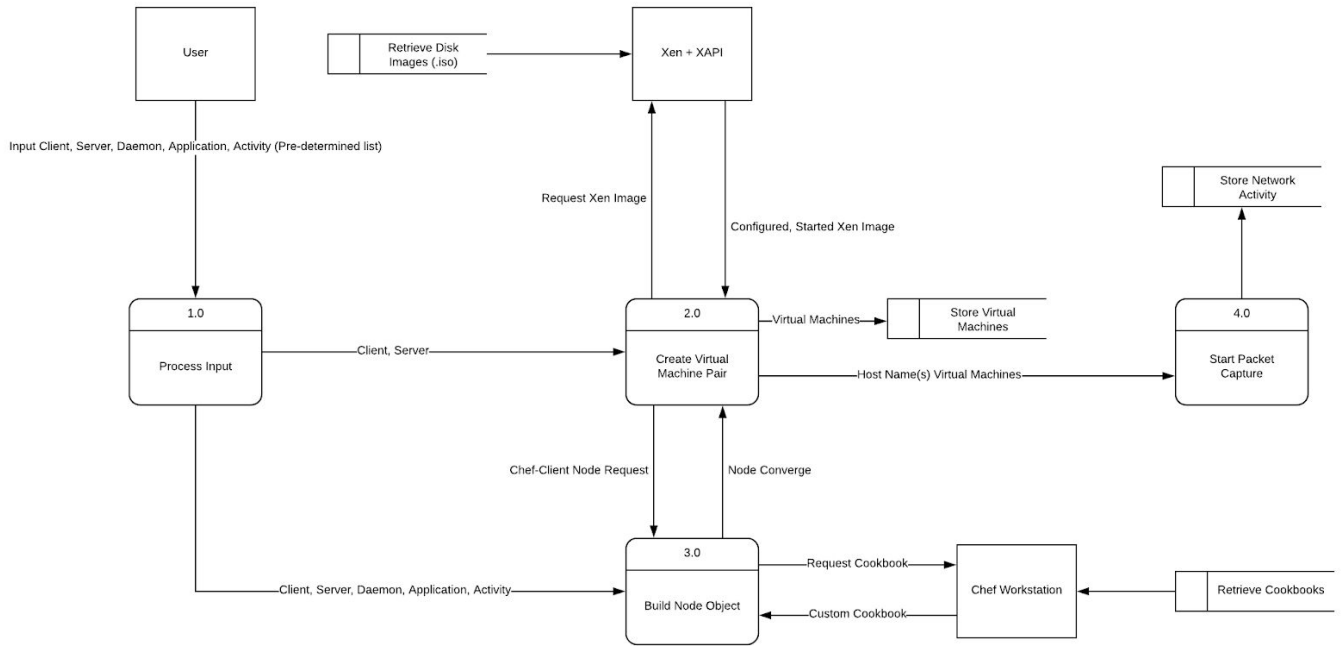
Figure 2: Data Flow Diagram Level 2 for Application

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.
.

2.2 Functional Requirements

1. **Ubiquitous Requirements**
   1.1. The hypervisor software shall be remotely accessible through a web application
   1.2. The web application shall provide secure user authentication prior to access
   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: Process 1 Input)
   1.4. The generated network flow shall be stored in a database for later access (Figure 2: Process 4 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 100 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
- ● Safety
  - ○ All hardware should be stored and operated in a safe and responsible manner
- ● Standard Protocol
  - ○ PEP8-compliant source code as a Git repository
- ● 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

- Python 2.7, 3.7, JavaScript
- Database systems
- Lightweight server deployment
- Web development
- Selenium
    - Web Scraper
    - Web Crawler
    - URL redirects/Web testing
- VMware/VirtualBox virtual machine experience

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 seperate 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 analysis device
    - Sends copies of data
    - Costly
    - License upgrade costs
- Vswitch for VMWare and Openvswitch for Virtualbox/Xen
    - Clean install
    - Flexible
    - Costly
    - ESX license
    - Physical switch for RSPAN and ERSPAN capabilities

Difference from market products
- Completely free
- Based entirely on open source material
- Application that will combine resources to automate traffic capture
- Generate virtual traffic automatically with prewritten automation scripts

2.5 Proposed Design

Possible Solutions:
- Virtual Machine
    - VirtualBox: Too buggy for projects that take a long time
    - VmWare: License requires a purchase
    - Xen: Too much flexibility
    - KVM: Not prefered

We went with Xen because it is flexible and closely relates to our project.

- Generate Traffic
    - Puppet: Doesn't include all features for our project
    - Chef: Has everything our project requires.
    - Ansible: Buggy
    - Expect Scripts: Not preferred

- Traffic Capture
    - PCAP: Record everything in the traffic.
    - NetFlow: Records only IPfix

We went with both, because we thought it would be nice to have multiple options in the traffic generator software

2.6 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 also requires a web application. Our team has a lot of web application development experience. Therefore, we came to a fast conclusion of using an MVC framework with HTML, JS, and CSS. There is 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 seemed enthusiastic about using Ruby for all of our scripting. Blakely also did not mind Python in Ruby's place. Therefore, 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. However, there were a lot of bug issues with VirtualBox that we did not think was worth the risk. 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. Dr. Blakely had also stated that he had experience with Xen.

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.

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 Xen API 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 suites 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
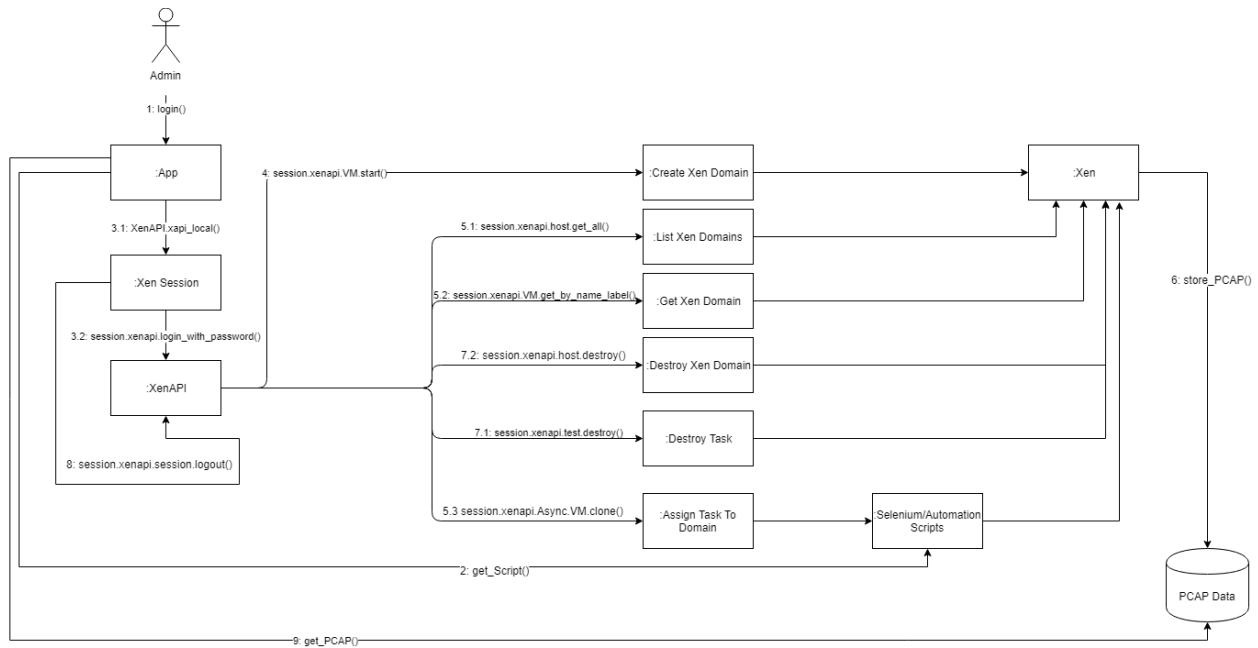
Figure 3: Communication Diagram (Iteration 1) for Xen

This figure describes how Xen and the Xen API will be interacting with the front end web application and database through specific calls.
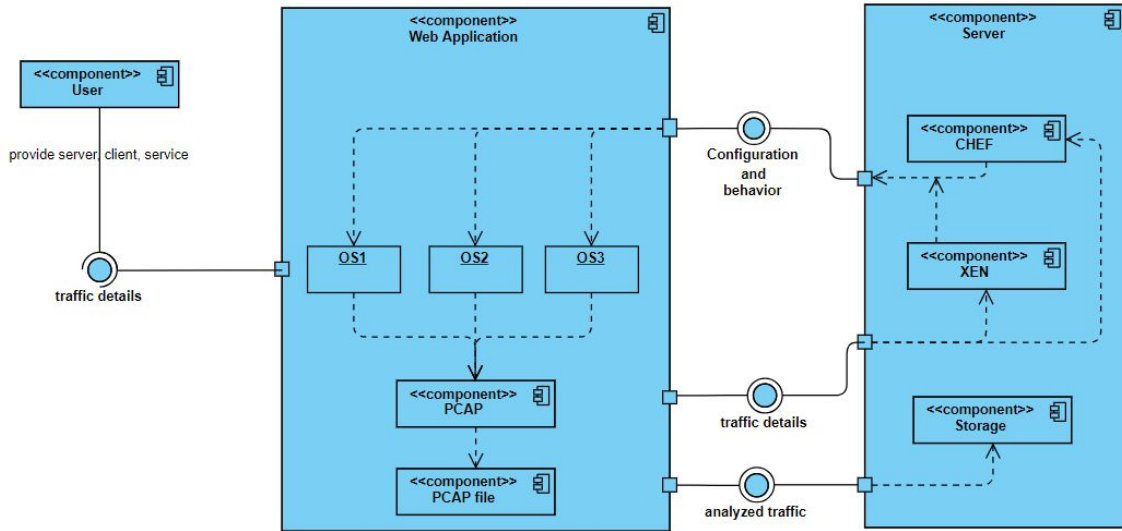
Figure 4: Component Diagram (Iteration 1) for Application

This figure shows how all components of our system fit together and where they will reside.
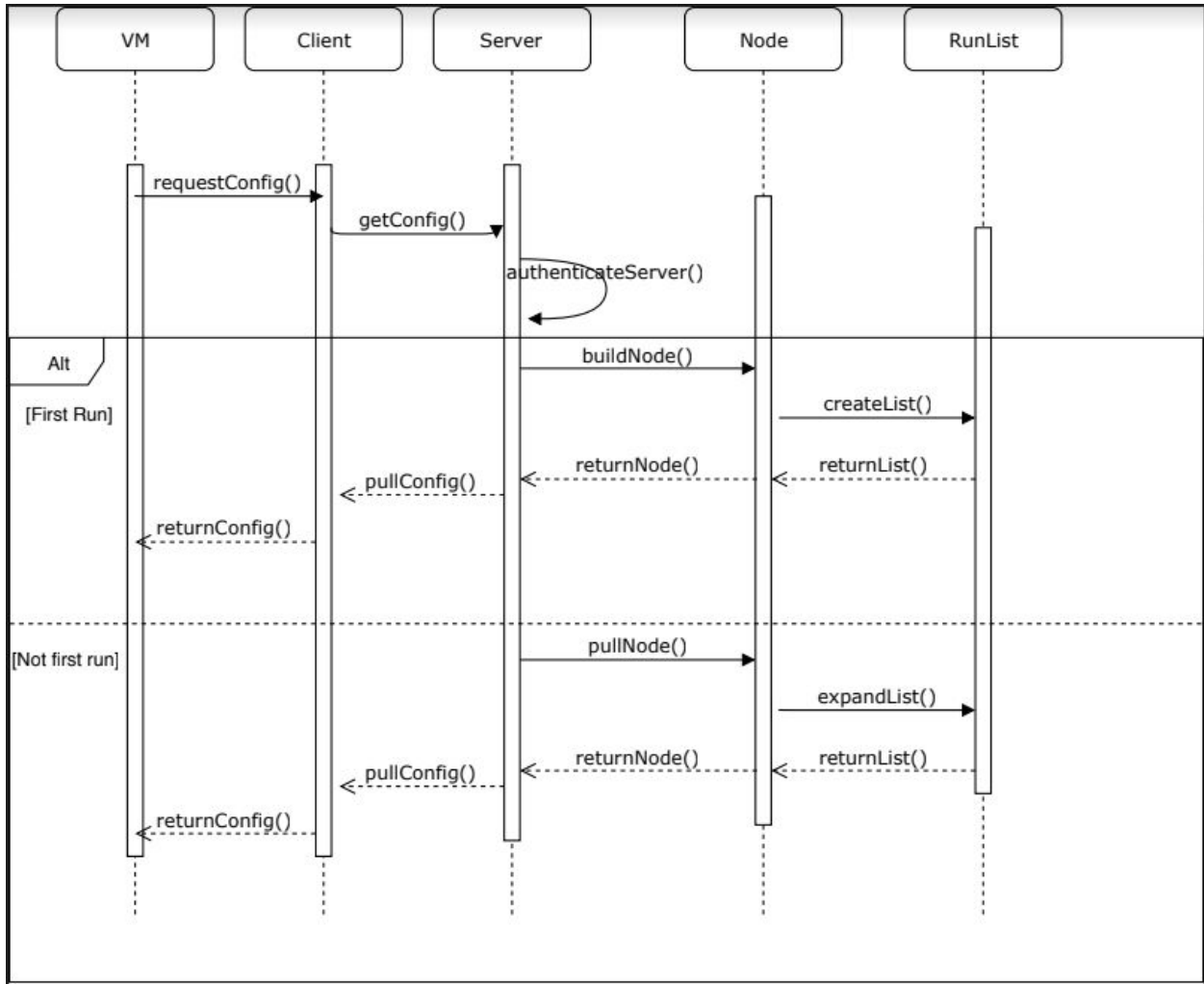
Figure 5: Sequence Diagram (Iteration 1) for Application

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

Figure 5: Use Case Diagram (Iteration 1) for Application

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

2.9 Possible Risks And Risk Management

Risks

- Incompatibility of software involved
  - Xen needs to handle applications and scripts launching automatically
  - Chef needs to work with Xen to automate configuration
  - Front end application will need to handle all use cases and properly communicate with virtual network, server, and server to database
- Hardware limitations
  - May not be able to hold as many scenario data as we would like
  - May not be able to launch complex virtual networks
- Time constraints
- Lack of expertise in virtualization of a network

Risk Management Strategies
- Communication in development
  - Slack as main conduit for the development team
  - Properly documenting and commenting on developed code
 - Proper use of GitHub
- Sharing expertise and knowledge
  - Code reviews
- Shrinking prototype executables to fit hardware limitations - or obtain more or better hardware
- 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.

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

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.

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
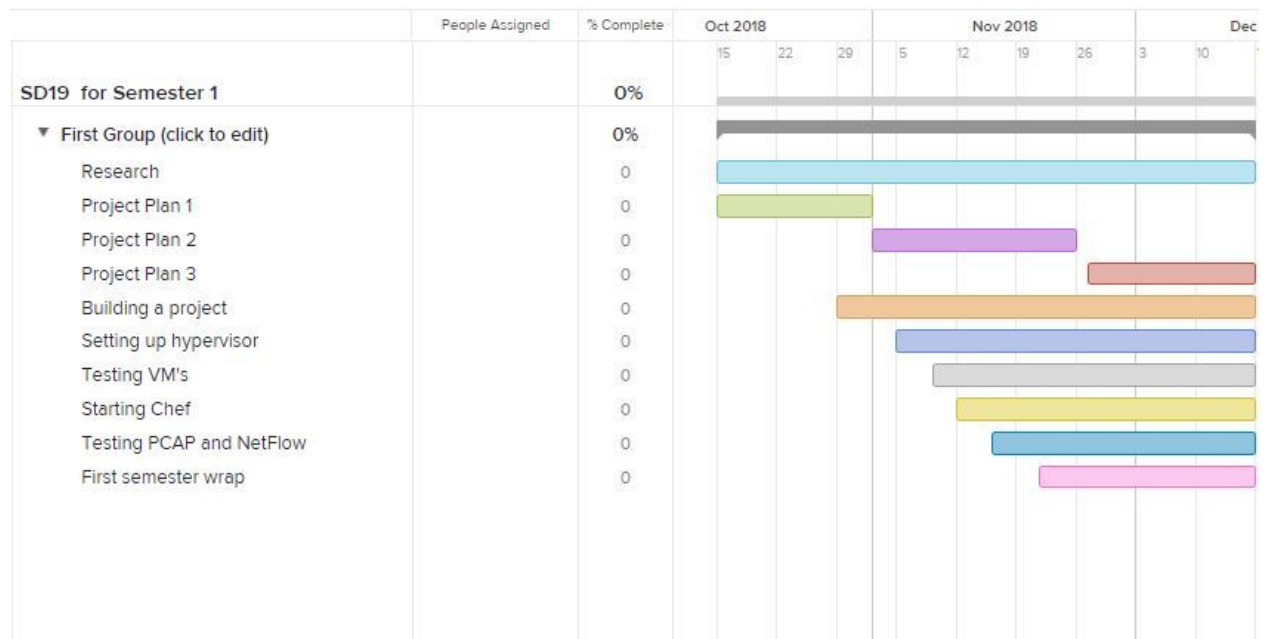
Figure 7 : Gantt Chart for first semester

- 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

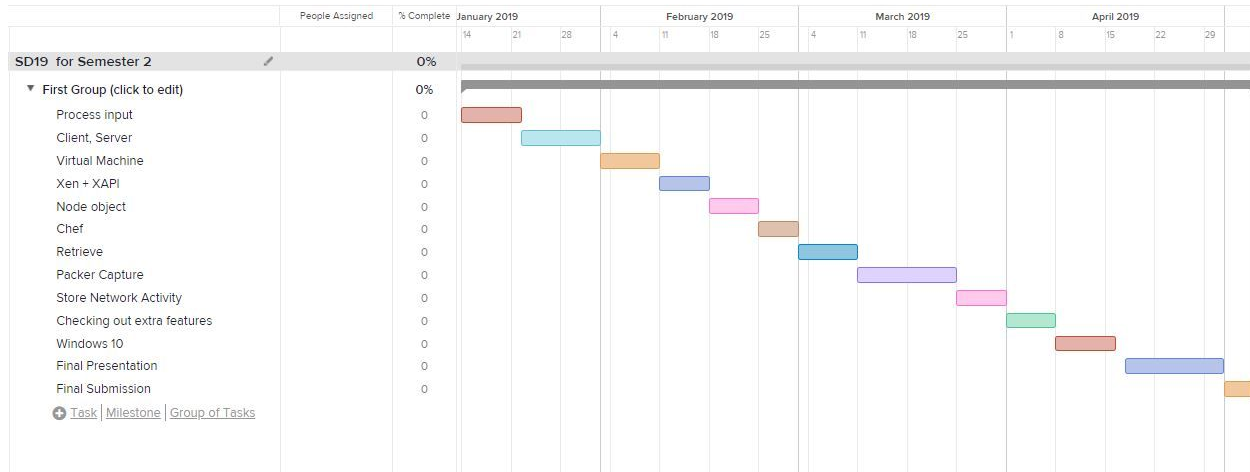|	People Assigned	% Complete	January 2019			February 2019			March 2019			April 2019		
			14	21	28	4	11	18	25	4	11	18	25	1	8	15	22	29
SD19 for Semester 2		0%																
▼ First Group (click to edit)		0%																
Process input		0																
Client, Server		0																
Virtual Machine		0																
Xen + XAPI		0																
Node object		0																
Chef		0																
Retrieve		0																
Packer Capture		0																
Store Network Activity		0																
Checking out extra features		0																
Windows 10		0																
Final Presentation		0																
Final Submission		0																

Task | Milestone | Group of Tasks

Figure 8 : 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 sends 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

The project is to create a tool that will enable cyber researchers to generate a specific type of traffic, capture and store the generated traffic, and analyze it in entirely automated manner. We

will follow the timeline mentioned in section 3.1 above, as we believe that this timeline will meet the requirements for implementing the web application tool.

3.3 Personnel Effort Requirements

Major Requirements:

Task	Description	*Estimated Time Required*
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
Testing Virtual Machines	Come up with several test cases or scenarios that we can run on the virtual machine according to our needs	30 hours
Testing PCAP and NetFlow	Come up with several test cases or captures that can 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	5 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	10 hours
Setup Chef	Setting up Chef to work in ways that fits our needs and requirements	10 hours
Develop Frontend	Developing the frontend of the project, such as the layouts and the webpage using various libraries .	60 hours
Develop Backend	Developing the backend of the project, such as integrating the automation of the virtual machines	90 hours

	that will be used in our project.	
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. 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. 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.

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
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
6. https://www.seleniumhq.org/
7. http://www.tcpdump.org/manpages/pcap.3pcap.html
8. https://www.solarwinds.com/what-is-netflow

4.3 Appendices

All diagrams were used in explaining the parts