Cyber Network Capture Generator

Design Document

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:

List of Figures

List of Definitions

ETG	*Electronical Technology Group*
GPL	*General Public License*
VM	*Virtual Machine*
ISEAGE	*Internet-Scale Event and Attack Generation Environment*

1 Introduction

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 and Project Statement

General 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. This project can be used to easily track malwares and bad traffic running through a network or application.

General Solution

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. The team hopes that we are able to come up with an web interface on the front end, with scripts and a program that will automate VM creation to test multiple scenarios that will be specified by the administrator, resulting in a printout of PCAP/Netflow files to show traffic.

1.3 Operational 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 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:

- The project is cost free
- PCAP files are very large that we might run out of space
- The tool will be used for research purposes only

1.6 Expected End Product and 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. Specifications and Analysis

2.1 Design Specifications

Functional Requirements

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: Process 1 Input)

1.4. The generated network flow shall be stored in a database for later access (Figure 2: Process 4 Output)

Event-driven Requirements

1.5. When the user selects client/server combination the hypervisor shall allocate and create two separate virtual machines (Figure 2: Process 2 Abstraction)

1.6. 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)

1.7. 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)

State-driven Requirements

1.8. While the virtual machines are active the server shall store network traffic to database (Figure 1: (OUT) “capture,” adjacent system)

Unwanted Behaviour Requirements

1.9. If hypervisor detects insufficient resources then the web application shall restrict virtual machine creation and display “insufficient resources” warning

1.10. If server detects insufficient storage then the web application shall display “insufficient storage” warning

Complex Requirements

1.11. When the network traffic is being stored, if the server detects insufficient storage, then the web application shall display “insufficient storage warning”

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 additional 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
- Standard Protocol.
    - PEP8-compliant source code as a Git repository.
    - CAPEC for maintaining the catalog of common attack patterns and vectors
- Ethicality
    - All work should be original for our development team with credit given to proper sources.
    - No unauthorized copying of software.

2.2 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

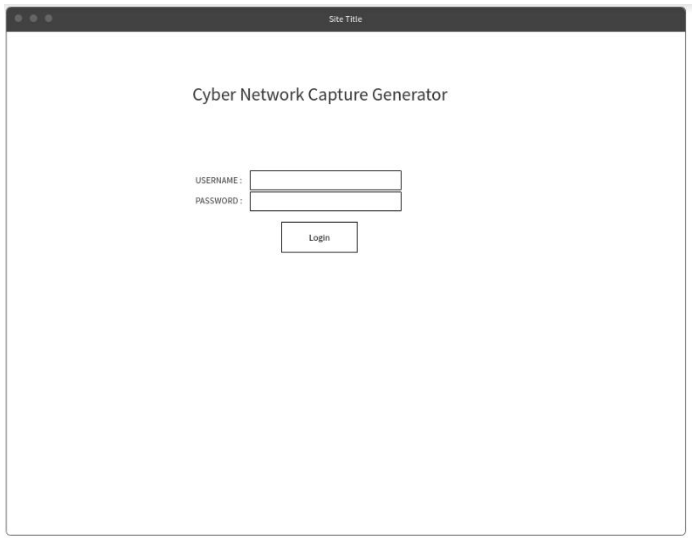


Figure 1 : Wireframe for Login Page

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

- Build Page

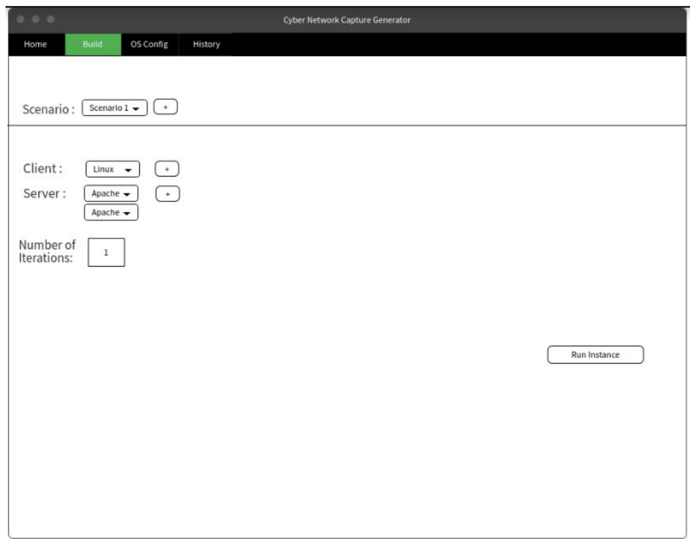


Figure 2: Wireframe for the Build Page

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

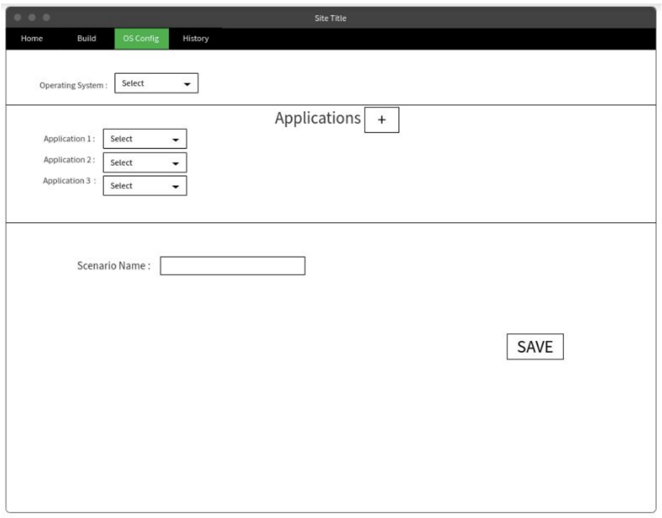
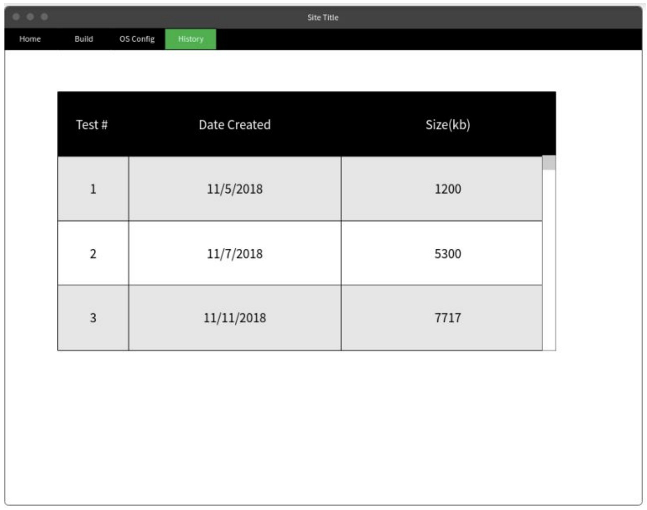


Figure 3: Wireframe for Creating a scenario

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

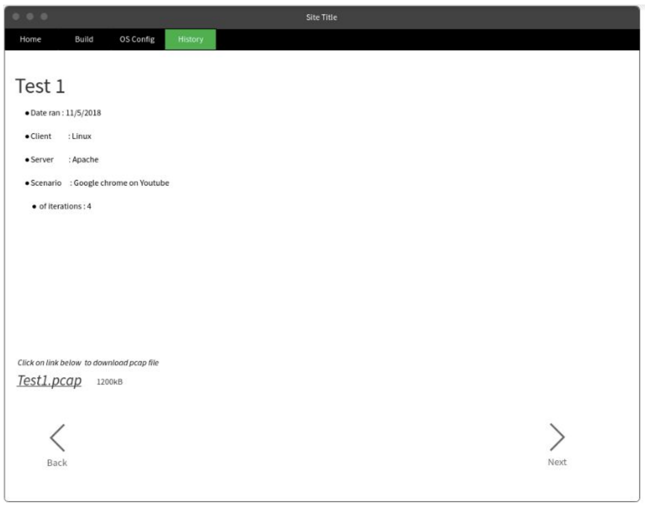


Test #	Date Created	Size(kb)
1	11/5/2018	1200
2	11/7/2018	5300
3	11/11/2018	7717

Figure 4 : 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



Test 1

- Date ran : 11/5/2018
- Client : Linux
- Server : Apache
- Scenario : Google chrome on Youtube
  - of iterations : 4

Click on link below to download pcap file

Test1.pcap 1200kB

Figure 5: Wireframe for Displaying Details of each test

This page can be accessed by clicking the number of the 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. Clicking the next or back button will bring them to the next or previous test accordingly.

Back-end:

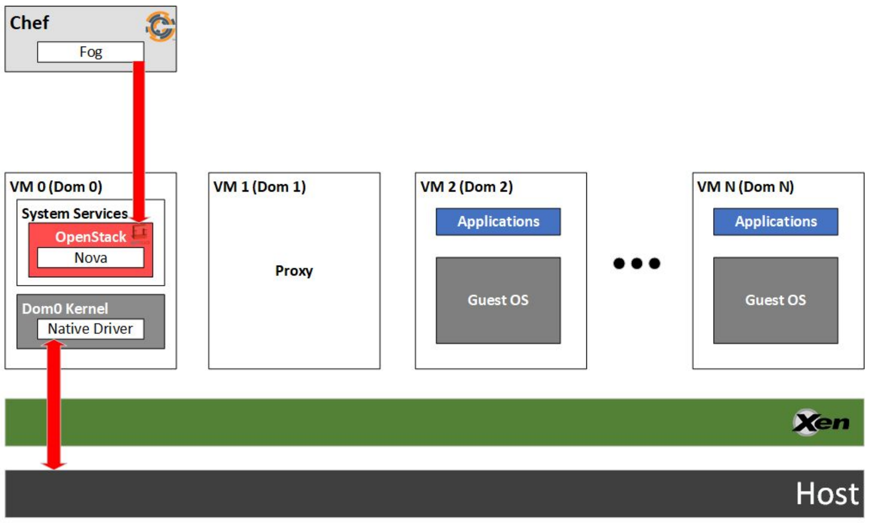


Figure 6: Backend system architecture

The proposed architecture in figure 6 relates to the creation of the virtual machines for the client and server machines. In order to do this several open source technologies are used, namely: Chef, OpenStack, and Xen.

Xen is a bare metal hypervisor. It has a secure and large structure that allows multiple operating systems to be run on the host system using virtualization techniques. In addition, for the networking aspect of our project it has a clearly defined networking architecture. This will allow for clearly defined virtual bridge, virtual lans, virtual routers between our virtual machines. This is important for the networking aspect of our project.

OpenStack, more specifically, Nova Compute is the controller for the hypervisor. This will provide the following tasks for our project: instantiating virtual machines, managing storage, and connecting to automated deployment service. Some of these tasks are interchangeable with

Xen Toolstack (xl), however, OpenStack is widely supported and opens up the ability to use automated deployment services.

Chef Provisioning will allow us to bootstrap created virtual machines. This will allow for the ability to automate the process of application installation, setup, and initial start.

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.

Further on our network strategy of using OpenWRT as a Dom in Xen as a “gateway” (more specifically router device) will route all traffic between LAN devices and another network WAN port acting as a gateway. We will be using OpenWrt as a Xen DomU guest. After the image is loaded and configured in Xen we will bridge the dom0 guest’s physical interface and a virtual interface that connects to the domU guest. The result is the domU guest receiving a connection on the same network as the dom0 guest. Then we will create two bridges for LAN and WAN. The LAN will only be accessible from the dom0 itself, while the WAN of domU will be bridged to the physical interface of the dom0 to obtain access to the internet.

2.3 Design Analysis

What we have done so far

Front-end:

Team members are new to front-end development. A large portion of time has gone into research and playing around with different builds of django framework. It is highly customizable as team members have learned, and time has been delegated into reading appropriate documentation. Developers have managed to establish some dummy routes and connect to a mock database. In addition, time has been taken to research the best possible web server. Research has been done in to both apache and nginx.

Back-end:

Team members in the backend have several different concepts to tackle. No individual team member can be involved/understand each part in the time period. So efforts have been made to appropriately divide the work to team members who have most interest in concepts. This can be thought of as: Virtualization, Automated Deployment, Networking, and Scripting. Team members have created models, performed tests (although broken), and have generated notes that will eventually turn into documentation.

Did it work? Why or why not?

Front-end:

Some success has been made in the way of generating “Throw Aways” of possible views. This has been beneficial in that it has made the team have to think about the function of the product as a whole, and brought the front end closer to a presentable product.

The developers in the front-end are still very new to full-stack development. Concepts of a route, database connection, etc. are still being developed. This has caused a lot of confusion in terms of setting up the web server. However, progress is being made.

Back-end:

Setup of the base box (that will become the server for the application) could be considered a success. An “alternate” form of Ubuntu 18.04 has been installed that allows for “LVM” which will speed up Xen as it creates virtual machines that will utilize computer volume. In addition, research has been made into isolating this behavior to its own disk partition for better security.

The back-end has had a lot of failure in terms of implementation. Attempts have been made to mix open source projects that do not work together. This has caused a lot of time to be lost.

However, through this considerable progress has been made in the way of coming to a working design.

Theoretically the back end has made good progress. Team members have openly discussed potentially at risk parts such as script, automated deployment, etc. and how they could lead to failure if not made with maintainability in mind. However, this is still very new to team members.

What are your observations, thoughts, and ideas to modify or continue?

Front-end:

Django is a large framework. To properly configure and set up with security in mind will take time. Team members still need to read proper documentation.

The front-end should have a simple interface. However, team members are not familiar with libraries that can be used to leverage a web site with best practices in mind. Time will have to be taken to research good libraries to best possible libraries to make the application simple yet professional.

The web server is still a fairly new concept. Research will have to be taken to properly configure apache. The routes and database will also need to transition to model the eventual backend structure as closely as possible.

Back-end:

Open source libraries come in many varieties. In order to find the correct libraries for a project takes a lot of upfront research and testing. Otherwise, libraries may not mix well. This leads to wasted time and headaches later down the road. It is better to front load research to mitigate this where possible.

Creation of the virtual machines is coming to a point where team members can better divide efforts. Moving forward, team members will split focus and test more tangible each of the concepts: networking (proxy, sniffing) on Xen machines; Custom machine generation with Chef Cookbooks; Dynamic cookbook creation; Traffic generation with Selenium browser scripts.

3 Testing and Implementation

3.1 Interface Specifications

Our project considerations will be almost entirely software based. Dealing with virtualization, web application, and a database; the only hardware interaction will be hosting the software. Because of this we will have four main interface specifications:

1. Web application UI.
    a. This is the interactivity between the user client and the rest of the system through the centralized web application.
    b. This interface will require ease of access in a user friendly fashion.
    c. This interface should only be used by authorized and authenticated users
    d. User will be able to provide input for client, server, daemon, application, and activity based on predetermined lists.
2. Framework interface with Xen.
    a. Xen images can be requested, configured, and started through this interface through calls to XAPI.
    b. Among these will be predetermined Selenium scripts and potential malware that can be loaded into the virtual environment.
    c. Virtual machines can be stored and accessed
3. Framework interface with Chef.
    a. Client, server, daemon, application, and activity input will be used in tandem with saved Chef cookbooks to request and retrieve custom cookbooks.
    b. These cookbooks will be used to create a node object to create a virtual machine pair.
4. Framework interface with Database.
    a. Handle the storage and retrieval of PCAP data gathered from the virtual environment

3.2 Hardware and software

Hardware

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. These desktops will help us to run multiple VMs without using our own personal computers. We will also be able to keep them running for a very long time.

Software

We will be using multiple softwares. They are categorized as follows :

- Xen
  - Xen is a hypervisor that uses microkernels to provide services that allow multiple operating systems to run on the same computer hardware. The main reason that Xen is a good choice as our hypervisor for the project is that the Xen project is fully free and open sourced and it is one of the requirements from our client. Also, Xen has good scalability compared to the other hypervisor choices. Xen also does well in separating the hypervisor execution from management OS,management stack,device drivers and it’s guests(components) which makes it more efficient.
- Chef
  - Chef is an open source configuration management software that will help in the automation of our environment. Chef is made up of different cookbooks and recipes that allow for system configuration in either on-premise of virtual environments. Chef seems to be one of the most flexible configuration management tools out there. This will be incredibly useful in the setup of our project’s environment as hopefully it will help automate the system setup process for each test run.Chef will utilizes a test-and-repair system for changes. This means that chef will only make changes to our system if one of the systems diverges from its assigned recipe/resource.
- Apache

    - Apache is the most widely used web server software. Developed and maintained by Apache Software Foundation, Apache is an open source software available for free. It runs on 67% of all webservers in the world. It is fast, reliable, and secure. It can be highly customized to meet the needs of many different environments by using extensions and modules
- Github
    - Github is widely used for maintaining source code and keeping track of work done.It's space is provided by the ETG department from Iowa State University
- Notepad++
    - An easy to use text editor and source code editor. Notepad++ is provided for free and can be used and redistributed under the terms of the GNUGPL.

3.3 Functional Testing

The following will be the tests that will be ran after the creation of the program :

Unit Testing:

- Mock database, through Django, Python "unittest", to load fake PCAP data
- Mock objects, through Django, Python "unittest", to load fake client/server model

System Testing:

- Python code is PEP-8 complaint.

Acceptance Testing:

- Server has defined host, application, and service installed
- Client has defined host, application, and service installed
- PCAP is captured, filtered by protocol, stored, and accessible through application.
- Application should be accessible through a web interface.

3.4 Non-Functional Testing

The following will be the tests that will be ran after the creation of the program :

Performance Testing:

- Web application should be able to remotely access the server in timely manner.

Scalability Testing:

- Web application should be able to initialize five virtual machine combinations for packet capture.
- Web application should be restricted from surpassing maximum capacity of server.

Security Testing:

- Web application is locked to specific accounts
- Database password are not in plaintext
- Network traffic is routed through ISEAGE, and any internet traffic makes use of proxy.

Usability Testing:

- Web application input fields for creation create virtual machine.
- Captured network traffic is accessible through web application.

Compatibility Testing:

- Server should be runnable on multiple box configurations.

Maintainability Testing:

- Server setup should be documented and replicable.
- Design decision, models, and interfaces should be documented for maintaining the application after passing on the software

3.5 Process

Testing

Currently, a large portion of the testing criteria is untested due to an emphasis on research for the project. However, there are several exceptions to this, including:

Unit Testing:

- Front end developers have instantiated mock databases using sqlite with theoretical data that will be used for scenarios and loaded from the backend.

Security Testing:

- Effort has gone into researching how to contain the ISEAGE environment into a Xen virtual machine. This would allow us to use it as the proxy requirement for our project. However, this is still theoretical. Routing traffic through the virtual machine and converting it from a VMware image to a Xen Image still need to be implemented.

Compatibility Testing:

- Xen, the hypervisor for our project, satisfies this requirement. Utilizing a concept known as "paravirtualization" the hypervisor and its subsequent images are able to run on even very old hardware.

Flow Diagram:

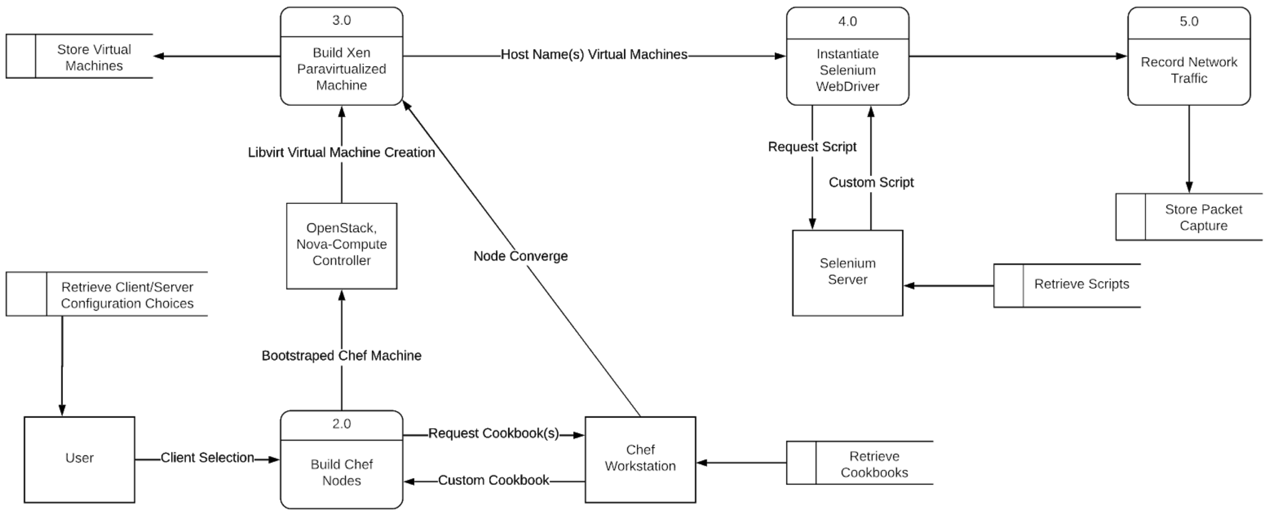


Figure 7: Data Flow Diagram Level 2 for Application

The flow described in figure 7, above, depicts the process necessary to generate network traffic from user's selected choices. The flow begins when a user inputs several choices that will define: setup of client machine; setup of server machine; steps to generate traffic. The "setup" is described steps 2.0,3.0 by utilizing three separate open source projects to bootstrap a virtual machine with chef-client, run the machine on the host system, and install the necessary applications. The steps for "generating web traffic" are somewhat ambiguous, and is still in the research stage.

3.6 Modeling and Simulation

For modeling and simulation, this project will have a framework (Django) that will be used in all simulations of the front-end development and ensuring that the front-end works as expected with no issues whatsoever.

For the back-end, an Apache server will be installed on top of the Django framework, creating a double layer that will be useful in creating a mock database for when we decide to model and simulate features from our lowest layer (Hypervisor) to the highest layer (Interface).

3.7 Implementation Issues and Challenges

Front-end:

One of the main issues we face in this implementation is building Apache on top of Django framework. Django works independently very well, and adding an Apache server on top of that proved to be one of the most difficult things we had encounters with. First, Apache had to be modified so that it connect directly to the Django framework. Then, the Django framework will connect to the server, and the apache will copy that connection from the framework. Finally, the server will start receiving and sending requests to the back-environment, which does all the work of the project.

Back-end:

Several challenges have been encountered in the back end of the project, including: networking concepts, virtualization concepts, and multiple large scale open source projects.

Networking is a large aspect of our project. From a learning aspect our team is very interested, however, all of the team members have either no or introductory experience. So tasks like implementing a proxy in the project to contain malicious traffic are more daunting. In addition, how to properly route the traffic so it can be effectively recorded is a challenging concept. A large portion of time has gone into researching how this might be accomplished. However, is still somewhat ambiguous. Our initial challenge was taking the Iowa State funded iseAGE project and applying it to our project for use as a proxy solution. This proved difficult, after contacting the monitoring professor Dr. Doug Jacobson for access and help we were referred to a spin off project iseRINK, a smaller, compact, and mobile version of iseAGE. From this we had to convert a completed esxi setup to xenserver to be compatible. Online tools such as starwind v2v converter were unable to complete this conversion process and maintain usability of iseRINK. In addition iseRINK's developed purpose is for cyber defense competitions, and as such its configuration is massive and intended for creating lab environments where students learn about penetration testing, networking, and other cyber security aspects. From here we moved to look for other proxy solutions. This involved research into different solutions: Squid and Streisand as potential proxies, securityonion for a security stack, pfsense for firewall solutions. Upon implementation attempts with Squid and Streisand we ran into lots of problems, mainly as Squid is a proxy to be used for web hosting and a lack of documentation between the two our backend team came across openWRT. openWRT is has much more flexibility in its design as a possible switch, router, or gateway with complete documentation on its deployment on a dom in Xen, which option to use, and how to implement its use. Even with all this documentation our team must familiarize ourselves with lengths of technical jargon for proper implementation.

In this project a main host computer must host several virtual machines. To do this the project will need to make use of a hypervisor. Properly configuring the "network" being developed on Xen VM's requires knowledge of routing and bridging VLANs. This is a rather difficult task for students with limited virtualization and networking knowledge.

System design in such large scale is challenging for students with limited experience. This has led us down several incorrect paths. For instance, at the start of the semester the team had decided several technologies it would like to use, namely: Xen, Vagrant, and Chef. However, this has been a headache. Vagrant once supported Xen by using a plug-in. After a lot of research and continuous attempts to use different approaches we learned this was true,at some point, but Xen was no longer supported. Not wanting to give up a lot of time went into possible ad-hoc solutions. This was a bad choice. The system had to be re-designed after continued failure, which has set back the back-end development. Attempting to learn from our prior mistake a lot of research has been front loaded into reading guides, documentation, etc. that made us believe the new direction was a true fit for the project.

3.8 Results

Results:

- Success: Created basic views that will turn into the website. Have spoken with client and started line of communication to create more professional, satisfactory product.
- Success: Created mock database and routes that will transition to become the back-end of the application over time.
- Success: Back-end has found, in theory, strong applicants for the open source project that if used properly mitigate many of the risks the back-end faces due to lack of expertise.
- Success: A lot of time research has gone into Xen. This will allow for better leveraging in terms of networking, third party libraries, etc.
- Failure: Repeated front-end design changes due to misunderstanding of web server functionality.
- Failure: Repeatedly mistaken project structures using old tools provided by xen. XEND, XAPI, etc. that resulted in lost time and mislead research (partial difficulty with large, open source project.)
- Failure: Wrong tool used for automated deployment. Led team members down rabbit trail with failed research into concepts, libraries, etc. that are not compatible with Xen architecture.

Plan for Future Progress:

Front-end:

- Continue progress made with design iterations and communication with client.
- Continue studying capabilities of Django and leveraging security and good design practices
- Continue research into third party libraries to create more professional front end application.
- Continue creating stronger data models, routes, and transition to database that will be used for actual application.

Back-end:

- Establish connection between Chef Provisioning, OpenStack, and Xen.
- Create initial general Cookbook (verify functionality of automation)
- Create virtual machine hosting proxy on Xen.
- Begin networking configuration and testing
- Create initial Selenium Script (verify functionality of traffic generation)
- Create initial client/server pair, load chef cookbooks, execute selenium script, record traffic, and compress and store to database (Proof of Concept)

4 Closing Material

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. This would be a spot where users can fully automate a test scenario for their possible vulnerability. 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

1. Noble, B. (2018). *Running Xen: A Hands-On Guide to the Art of Virtualization*. [online] Barnes & Noble. Available at: https://www.barnesandnoble.com/w/running-xen-jeanna-n-matthews/1126808408 [Accessed 13 Oct. 2018].
2. Ewart, J. (2017). *Chef: Powerful Infrastructure Automation*. Birmingham, UNKNOWN: Packt Publishing.