Cyber Network Capture Generator

Design Document

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*

List of Figures

List of Definitions

ETG : Electronical Technology Group
GPL : General Public License
VM : Virtual Machine

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

Limitations:

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

1.6 Expected End Product and 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. Specifications and Analysis

2.1 Proposed Design

Possible Solutions:

- Virtual Machine
  - VirtualBox: Too buggy for projects that take a long time
  - VmWare: License requires a purchase
  - Xen: has tons of flexibility and, thus, we choose it as our hypervisor program
  - KVM: Not prefered

- Generate Traffic
  - Puppet: Works well with our project.

- Configuration Management
  - Chef: Has everything our project requires.
  - Ansible: Buggy
  - Expect Scripts: Not preferred

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

Done so far:

- Researching how to Virtual machines using Xen.
- Researching Traffic generation using PCAP and Netflow
- Researching Generating cookbooks using Chef.

2.2 Design Analysis

No centralized prototype has yet been developed. All observations made have been from individual experimentation with relevant softwares. Current tests with Xen on private machines are a proof of concept that virtualization is possible and practical through XAPI. No scripts or applications have been run, and no PCAP data has been monitored or captured yet. Our next team step for Xen is to load it up on our provided hardware and build a framework for interactivity from a developed front-end web testing applications. From this framework we will begin testing interactivity with Chef and a database.

Because our team has found no problems in compatibility or testing through experimentation no design plan modifications need to be made at this time. Our biggest strength is strong documentation in our project plan so that the team knows exactly what tasks need to be done and in what chronological order. Our weaknesses at this point include the lack of overall development of the system that is subject to testing. Further iterations will allow for further design analysis.

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

<u>Hardware</u>
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.

<u>Software</u>
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.
- 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

Currently, the team has not ran any tests as we are still in the research phase of the project. The proposed flow will be as the diagram below:

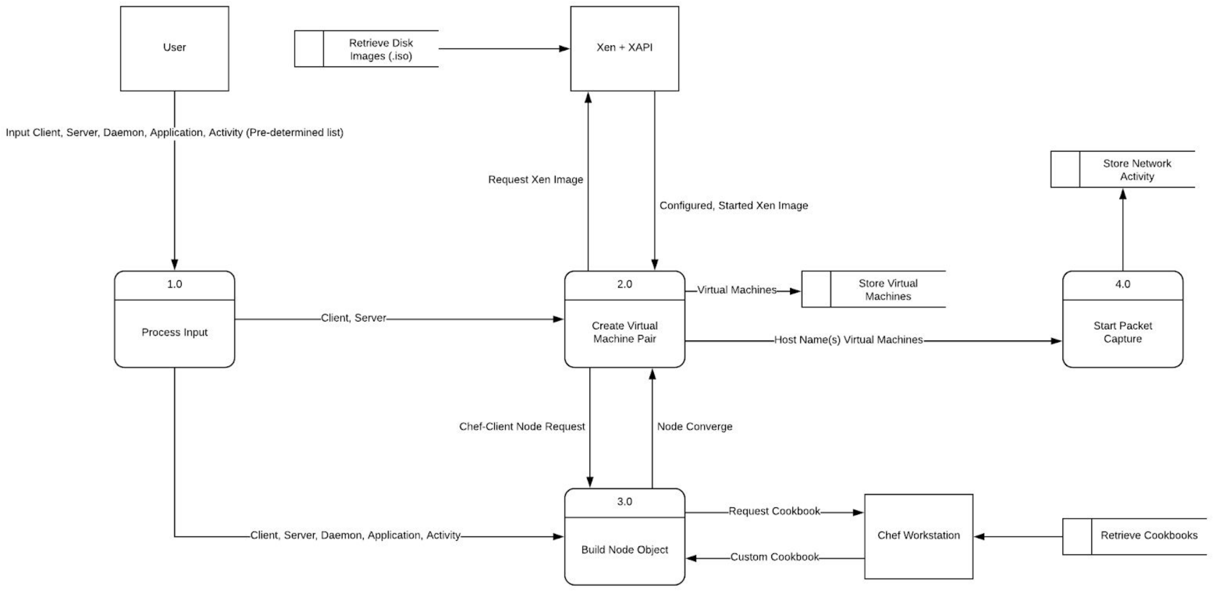

Figure 1: Data Flow Diagram Level 2 for Application

3.6 Results

So far this project has been mostly researching all possible solutions. Some members have tested with Chef and Xen. These tests have come to mostly be just tests in learning the softwares for each members. The next few weeks will be dedicated to the testing of integration of all of our proposed softwares and solutions.

4 Closing Material

4.1 Conclusion

After doing a lot of researching, we started to implement the project just now. Our goals are as follows:

- Create a computer software that is inaccessible to malicious attacks.
- Ensure that the software can handle multiple Virtual Machines at the same time.
- Correctly implement the “fake” traffic generation by PCAP and Netflow
- Integrate Cookbooks from Chef, such that Xen will automatically fetch the user’s requested Cookbook.
- Build a hypervisor or a Virtual Machine Monitor (VMM) that supports all Virtual Machine constructed.

This plan of action may seem far-sighted. However, we believe the best plan of action is one that includes all possible scenarios, and one that has to be as efficient as possible and as concise as possible.

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.