*The American University of Iraq, Sulaimani*

**Automated Malware Analysis in Virtual Reality**

Capstone Report

Hussein A. Shahab

Supervised by: Dr. Hemin Latif

1 Introduction 3

1.1 Background 3

1.2 Problem Statement 3

1.3 Aims and Objectives 4

1.4 Expected Outcomes 4

2 Project Planning 4

2.1 Resource Requirements 4

2.2 Budget Description 6

2.3 Project Time Schedule 6

3 Risk Management 8

4 Literature Review 9

5 System Design and Operation 10

5.1 System Architecture 10

5.2 System Operation 10

6 Virtual Reality Interface 11

7 Project Challenges 12

8 Future Work 13

1 Introduction

1.1 Background

One of the major and serious threats on the Internet today is malicious software, mostly referred to as malware. The malwares being design by attackers are polymorphic and metamorphic which have the ability to change their code as they propagate. Moreover, the diversity and volume of their variants severely undermine the effectiveness of traditional defenses which typically use signature-based techniques and are unable to detect the previously unknown malicious executables. The variants of malware families share typical behavioral patterns reflecting their origin and purpose. The behavioral patterns obtained either statically or dynamically can be exploited to detect and classify unknown malwares into their known families using machine learning techniques.

Malware Analysis is a very specialized field in Information Security that requires years of training to comprehend the most basic of today’s malware. Very few samples of today’s malware will stand against dedicated, individual attention by a team of analysts. The problems we face today are about processing the millions of samples that are disseminated every day. Many of these samples are duplicated or slight modifications, detecting which malware samples are similar is a big thing. So is the automated analysis of malware. These are both complex topics that use some advanced techniques, such as symbolic execution and machine learning.

1.2 Problem Statement

Automated malware analysis produces huge number of raw datasets for CPU registers, cache, memory dumps, registry keys, network traffic, activity logs, processes tables and kernel statistics all related to the execution of the malware sample in a sandboxed environment. It would require a highly skilled analyst to dig through these datasets in order to identify and correlate the functions, origins and the family variants of the malware sample. This project aims to tackle this issue by developing a VR engine with an integrated sandboxed environment for automated malware analysis (AMA) making the analysis phase more efficient and less time consuming. In addition to lowering the learning curve for the role of the malware analyst, less analysis skill and experience will be needed to come to the same outcomes as an AMA with skilled personnel.

1.3 Aims and Objectives

**Aim**

* Design, develop and evaluate a distributed VR system integrated with an AMA engine.

**Objectives**

* Acquire software and system requirements for the AMA engine.
* Design the system architecture for the AMA engine.
* Implement the AMA engine on a physical host server.
* Deploy 3 VMs on the AMA engine, serving as the sandboxed environment.
* Configure RESTful web service on the AMA engine.
* Test the AMA engine.
* Acquire software and system requirements for the VR engine.
* Design the system architecture for the VR engine.
* Develop the VR engine.
* Deploy the VR engine on a physical host server.
* Test the VR engine.
* Integrate the VR engine with the AMA engine through REST APIs.
* Test system integration.

1.4 Expected Outcomes

The expected outcome of this project is a 3D virtual environment serving as an interface for the malware analyst for interacting with the AMA engine.

2 Project Planning

2.1 Resource Requirements

To implement the project, the following resources will be required:

Hardware

1. Physical server hosting the AMA engine

[8GB RAM, ~60GB storage w/ gigabit network interface]

1. Physical server hosting the VR engine

[32GB RAM, ~120GB storage, GTX 1080Ti, 8-core processor w/ gigabit network interface]

1. Oculus Rift + Touch controllers
2. 2x Oculus sensors

Software

1. Parrot Security OS version 3.8
2. Windows 10
3. 2x Windows 7
4. CentOS Linux version 7.1708
5. Cuckoo Sandbox
6. PostgreSQL version 10.0
7. TCPdump
8. Volatility
9. M2Crypto
10. SWIG
11. Unity 2017.2 Plus
12. 3Ds Max 2018
13. Shader Forge
14. CurvedUI
15. VR Panorama 360 Pro Renderer
16. VRTK – VR Toolkit
17. PowerUI 2
18. Python 2.7
19. Python 3.6
20. C# 7.0
21. .NET 4.7
22. Allegorithmic toolset

2.2 Budget Description

The estimated budget for the project:

|  |  |
| --- | --- |
| **Item** | **Price** |
| AMA server | $1,000 |
| VR server + shipping | $2,499 + $514 |
| Oculus Rift & Touch + shipping | $499 + $119 |
| Unity Plus License/Year | $395 |
| CurvedUI | $25 |
| VR Panorama 360 Pro Renderer | $49 |
| PowerUI 2 | $90 |
| **Total** | **$5,190** |

2.3 Project Time Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Start Date** | **End Date** | **Duration** | **Activity** |
| 3rd October, 2017 | 5th, 2017 | 3 Days | Write the Project Proposal |
| 6th October, 2017 | 6th October, 2017 | 1 Day | Acquire software and system requirements for the AMA engine. |
| 7th October, 2017 | 9th October, 2017 | 3 Days | Design the system architecture for the AMA engine |
| 10th October, 2017 | 16th October, 2017 | 7 Days | Implement the AMA engine on a physical host server. |
| 17th October, 2017 | 18th October, 2017 | 2 Days | Deploy 3 VMs on the AMA engine, serving as the sandboxed environment. |
| 19th October, 2017 | 20st October, 2017 | 2 Days | Configure RESTful web service on the AMA engine. |
| 21st October, 2017 | 21st October, 2017 | 1 Day | Test the AMA engine. |
| 22nd October, 2017 | 22nd October, 2017 | 1 Day | Acquire software and system requirements for the VR engine. |
| 23rd October, 2017 | 25th October, 2017 | 3 Days | Design the system architecture for the VR engine. |
| 26th October, 2017 | 21st November, 2017 | 27 Days | Develop the VR engine. |
| 22nd November, 2017 | 22nd November, 2017 | 1 Day | Deploy the VR engine on a physical host server. |
| 23rd November, 2017 | 23rd November, 2017 | 1 Day | Test the VR engine. |
| 24th November, 2017 | 26th November, 2017 | 3 Days | Integrate the VR engine with the AMA engine through REST APIs. |
| 27th November, 2017 | 28th November, 2017 | 2 Days | Test system integration. |

3 Risk Management

The following risks are identified for this project:

|  |  |
| --- | --- |
| **Rank** | **Project Risk** |
| 1 | Delay of objective’s completion date due to other class’s exam/assignment |
| 2 | Delay of shipments due to international travel ban on Kurdistan |
| 3 | System requirements not accurately identified |
| 4 | Excessive schedule pressure that might reduce productivity |
| 5 | Lag of internet access in AUIS dorms |
| 6 | More optimization is required for the Real-time VE rendering |
| 7 | Low performance in RESTful web services |
| 8 | Virtualization hypervisor interoperability with the VR engine |
| 9 | Implementation of 3D interactive charts takes a long time |

Contingency plans for the identified risks:

|  |  |
| --- | --- |
| **Risk** | **Contingency Plan** |
| 1 | Avoid delay by shifting critical objectives start and end dates |
| 2 | Replace rift goggles with a VR emulator |
| 3 | Upgrade servers’ hardware |
| 4 | Focus on AMA and VR completion dates |
| 5 | Complete internet-related tasks off campus or while visiting Erbil |
| 6 | Reduce VE tessellations if time allows it or install a second GPU |
| 7 | Reduce API calls if possible or implement multipart batched request |
| 8 | Replace the hypervisor with ESXi or XenServer (Bare-metal) |
| 9 | Reduce interactive features for charts |

4 Literature Review

There is one known application of malware analysis in virtual reality that was revealed by Avast on September 14th, 2017 at the Mobile World Congress Americas. The demo showcased data visualizations pertaining thousands of malware samples for Avast’s cybersecurity network. An analyst can pick multiple samples and compare their memory dumps with one another to identify similarity in the way they function. Few applications are available in the market for data visualization in VR, such as: datavizVR, Kineviz. These applications are designed for general-purpose visualizations and not as an interface for other systems. Moreover, the field of malware analysis witnessed a huge number of developments partly due to governments backing and funds. Starting from Manual Code Reversing, where it took an analyst large amount of time and effort to reverse engineer the binaries, to Interactive Behavior and Static Properties Analysis, where an executable gets sandboxed inside the memory and observed while being executed, and reaching today’s Fully-Automated Analysis and hopefully reaching Fully-Automated Analysis in VR. Combining both malware analysis and VR is significant due to the difficulty in visualizing and identifying the generated malware’s datasets. Monnappa K. A. focuses on this issue in his talk at Europe Black Hat conference titled, “Automating Linux Malware Analysis Using Limon Sandbox” He talks on today’s malwares and how identifying becomes really problematic due to the way they hook, inject and/or hijack processes in memory without residing on the hard disk. This becomes even troublesome for digital forensics where an investigator can misattribute the origin and creation of the malware to a different country.

5 System Design and Operation

5.1 System Architecture

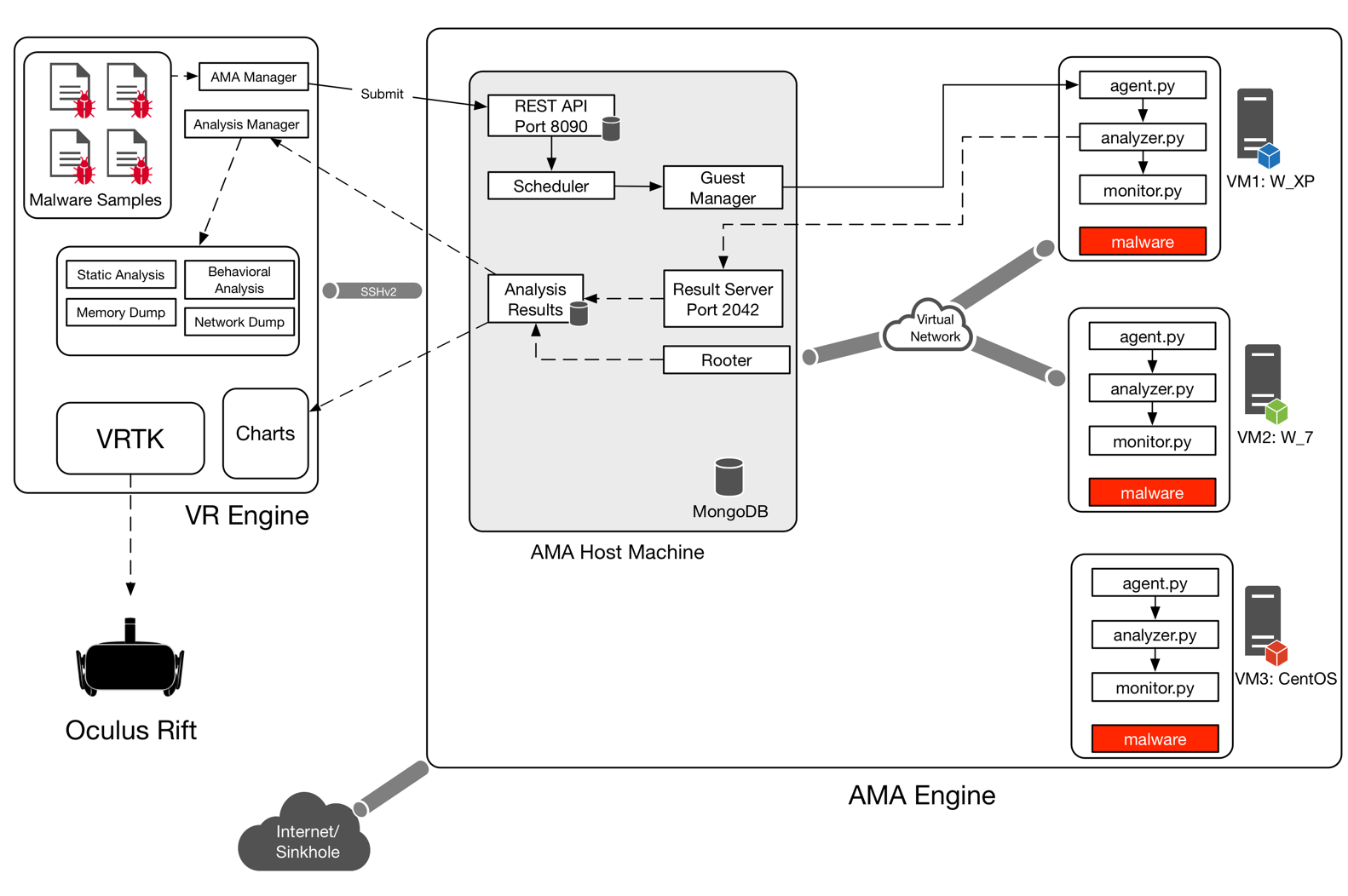


Figure 1 - System Architecture and Operation Diagram

5.2 System Operation

The system is designed to be mainly operated through the oculus rift and touch controllers. The operator is connected through VR goggles to the Security Operations Center environment inside the VR engine. The operator is surrounded by three panels at 270-degrees angle, from which the operator is able to browse through malware samples and submit them for analysis. The malware analysis part is done through creation of tasks, attaching a malware sample to the task, and submitting the task to the AMA Host Machine through the AMA Manager on the VR engine. Then, the task is scheduled to run in a sandboxed environment of 3 virtual machines. Once the scheduled task is ready for execution, the Guest Manager restores a snapshot for the selected VM to a clean state that is ready to execute the malware inside it. The Guest Manager communicates with the python agent to transfer the malware sample to the VM and prepares hooks for the analyzer from which malware execution behavior is collected and profiled.

The operation for the analysis system involves a custom Cuckoo Sandbox implementation acting as the AMA engine. The server is hosted on Linux OS running Parrot Security. The VR Engine establishes an SSHv2 tunnel when it is started, to prepare and run services that are needed for the AMA Host Machine, such as: MongoDB, Rooter, REST API, and Guest Manager. After the malware has completed execution, the analyzer collects all the static and behavioral analysis results in the Result Server. The Rooter collects and dump network traffic that is routed from the VM’s virtual network to the Internet/Sinkhole. The VR engine fetches all the results in JSON format through the Analysis Manager and parse these information into reports and charts inside the Security Operations Center.

6 Virtual Reality Interface

The interface is controlled through Oculus Touch controllers that are capable of capturing operator’s hand presence. Implemented gestures composed of communicative gestures, object manipulation, and UI navigation. All input mapping is accomplished through OVRInput handler for the Oculus SDK.



Figure - hand presence with fingers tracking

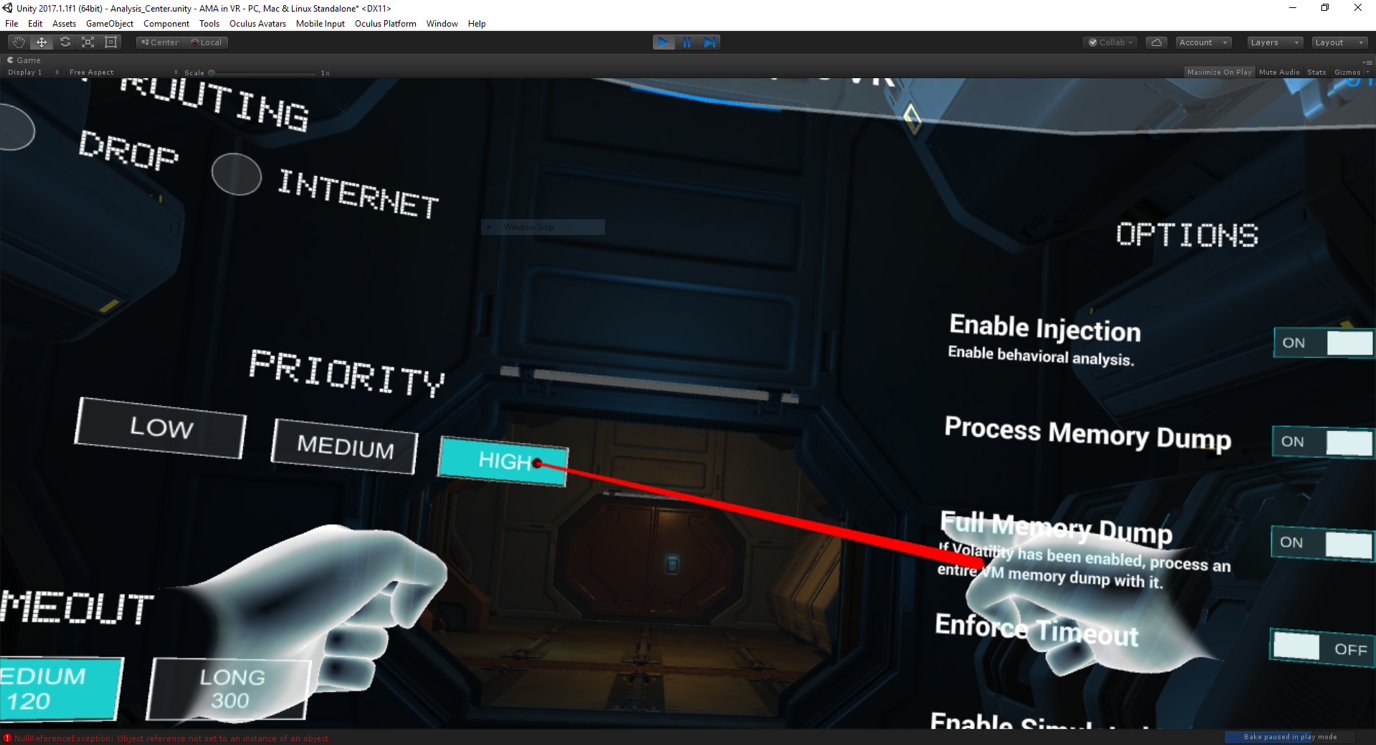


Figure - UI navigation through laser pointer

7 Project Challenges

Development of virtual reality applications is challenging and computing-intensive, since input handling and processing must be accomplished without affecting the application’s core performance. Performance hiccups are extremely noticeable due to the nature of virtual reality real-time interactions. Threading was used extensively during the development phase to maintain a minimum solid frame rate of 60fps. Project’s time schedule was frequently pushed back and changed due to other coursework assignments during the semester. Essential equipments for the project were delayed in shipping because of the imposed travel ban on KRG. Project’s cost was managed and paid all by the developer without company or university’s support due to the financial crisis. Project development was conducted in AUIS dormitories. Unfortunately, Internet access restrictions and performance delayed the deployment of the AMA engine by 1 week and other alternatives were used. Despite all the challenges, working on this project and overcoming them was extremely fun and challenging on many technical and non-technical aspects.

8 Future Work

Development for this project will definitely continue in the future. Features such as remote collaboration between operators, multiple background environments, performance tuning for low-end PCs, and further improvements for the UI and charts. Complete documentation for the source code and usage will be upload online for easy reference. Licenses will be added for integrations with commercial and third-party software for data visualizations, behavioral analysis, and artificial intelligence.