Fall 12-20-2013

Categorization of Large Corpora of Malicious Software

Deekshit Kura
dkura@uno.edu

Categorization of Large Corpora of Malicious Software

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science
Information Assurance

by

Deekshit Kura
B.Tech, Jawaharlal Nehru Technological University, 2011

December, 2013

Dedication

To my parents,

and to those who dedicate their lives

for the betterment of others,

expecting little to nothing in return.

Acknowledgement

I owe the deepest gratitude to my thesis adviser, Dr. Golden Richard III, for his encouragement, guidance, and support. His enthusiasm in the research area of malicious software has been motivational and made this thesis a pleasure. I am also grateful for the time and input contributed by my other committee members, Dr. Irfan Ahmed and Dr. Adlai N Depano.

I am very much thankful to Dr. Shengru Tu for giving me such an amazing and solid programming foundation in databases, distributed systems, concurrent programming and to Dr. Golden Richard III for their teachings in reverse engineering, and principles of operating systems II as these are the areas I plan to pursue.

I want to express my sincere appreciation to Dr. Irfan Ahmed for his guidance and help while keeping me in the right direction for this thesis.

I would also like to thank Mr. Danny Quist for malware collection. Without his large malicious software contribution, I would most likely still be searching for samples! Finally, I thank my family and my supervisor for their support and help throughout the duration of my thesis and coursework.

Table of Contents

List of Figures

List of Tables

Abstract

Malware is computer software written by someone with mischievous or, more usually, malicious and/or criminal intent and specifically designed to damage data, hosts or networks. The variety of malware is increasing proportionally with the increase in computers and we are not aware of newly emerging malware. Tools are needed to categorize families of malware, so that analysts can compare new malware samples to ones that have been previously analyzed and determine steps to detect and prevent malware infections.

In this thesis, I developed a technique to catalog and characterize the behavior of malware, so that malware families, the level of potential threat, and the effects of malware can be identified. Combinations of complementary techniques, including third-party tools, are integrated to scan and illustrate how malware may harm a target machine, search for related malware behavior, and organize malware into families, based on a number of characteristics.

Keywords: Malware Analysis

Chapter 1

Introduction

What is Malware?

Malware or malicious code (malcode) is short for malicious software. Malware is a combination of the words “malicious” and “software” and is a piece of software (computer program) written by someone with mischievous or, more usually, malicious and/or criminal intent. It is also another term for “computer virus.” “Virus” is the term most often used to describe computer malware.

It can also gather sensitive information without our permission or knowledge and, gain unauthorized access to the system resources. It can do some sort of damage or theft. It can cause serious damage to files in our computer. It hosts illegal data on infected computer systems, sending spam email, and attacking other machines. The majority of malware spread in our computers is through the internet.

Malware cannot damage the physical hardware of current generation computer systems and network equipment, but it can damage data residing on the systems and seriously impact available resources, by consuming RAM, network bandwidth, or storage spaces. Malware should also not be confused with defective software, which is intended for legitimate purposes but has errors or bugs.

Types of malware

Malware are classified into various categories and spread in different ways. These include:

- Viruses
- Trojan horses

- Worms
- Rootkits
- Spyware
- Bots
- Back doors, etc.

Three of the most common malware types are viruses, worms and trojans. These types of programs are able to self-replicate and spread copies of themselves, which might be modified copies. These categories of malware are described in more detail below:

Viruses

The term computer virus is used for a program that has infected some executable software and, when run, causes the virus to spread to other executables. It is a type of malware that propagates by inserting a copy of itself into and becoming part of another program. It spreads from one computer to another, leaving infections as it travels.

Generally, a virus cannot be spread without a human action, such as running an infected program.

Viruses are of different types, as follows:

- File viruses
- Macro viruses
- Master boot record viruses
- Boot sector viruses
- Multipartite viruses
- Polymorphic viruses
- Stealth viruses

Worms

Worms are malicious programs that make copies of themselves again and again on the local drive, network shares, etc. Unlike a virus, it does not need to attach itself to an existing program. Worms spread by exploiting vulnerabilities in operating systems and applications, taking advantage of file-transport or information-transport features on the system to allow them to travel unaided.

Trojans

A Trojan horse is not a virus. It is a destructive program that looks like a genuine application. Unlike viruses, Trojan horses do not replicate themselves but they can be just as destructive. Trojans are also known to create back doors to give malicious users access to the system, allowing confidential and personal information to be stolen.

Trojans essentially invite users to run it, concealing harmful or malicious code. The code may take effect immediately and can lead to many undesirable effects, such as deleting the user's files or installing additional harmful software. It can make any number of attacks on the host by, stealing data or, activating and spreading other malware.

The seven main types of Trojan horses are:

- Remote Access Trojans
- Data Sending Trojans
- Destructive Trojans
- Proxy Trojans
- FTP Trojans
- Security Software Disabling Trojans
- Denial-of-service Attack Trojans

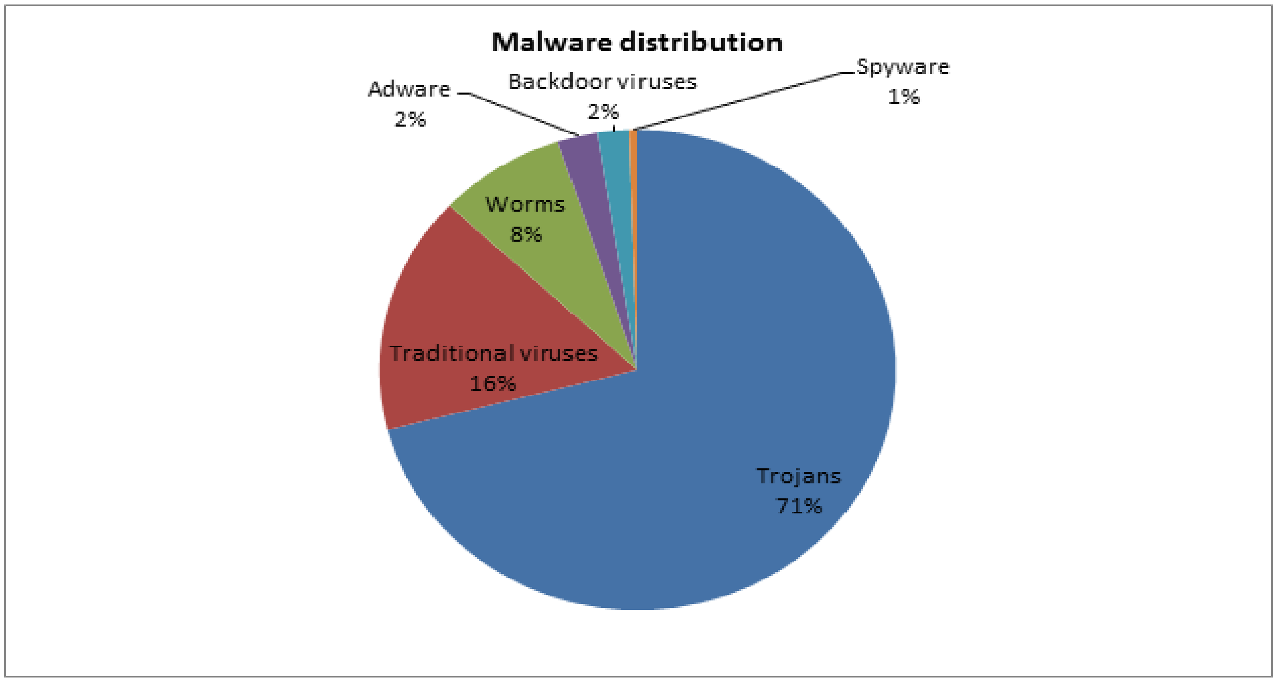
Spyware

Spyware is a type of program that is installed with or without user permission on a personal computer to collect information about users, including things like browsing habits or financial data, without user consent. Sometimes the data is collected for such reasons as advertising, but can also be collected to perpetrate identify theft or to abuse credit cards. It also can download other malicious programs from the Internet and install it on a computer.

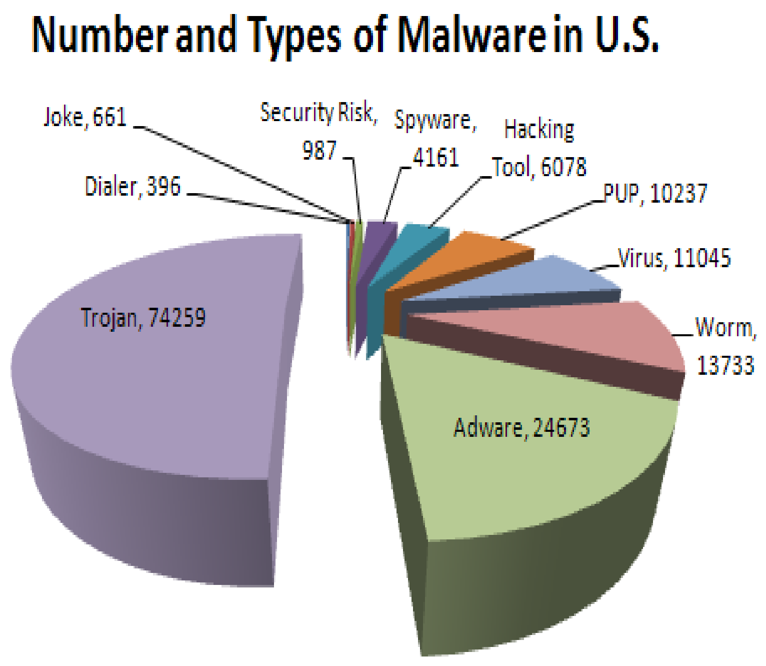
Backdoors

A back door is an undocumented way of accessing a system, bypassing the normal authentication procedure. Backdoors may also be installed prior to malicious software, to allow attackers entry and when once the system is compromised, one or more backdoors maybe installed to allow easier access to the systems.

The following figure shows the distribution of the malware in percentile and the types of malware present in the U.S.



Fig(a). Malware Distribution (Ref. 1)



Fig(b). Types of Malware in U.S (Ref. 2)

Chapter 2

Anti-malware software and best practices - Avoiding malware and viruses

Anti-malware programs can combat malware in two ways: (Ref. 3)

1. They can provide real time protection against the installation of malware software on a computer. Such software works by scanning incoming network data for malware and blocks any threats that are detected.
2. Anti-malware software programs can be used solely for detection and removal of malware software that has already been installed onto a computer. It scans the contents of the registry, operating system files, and installed programs on a computer and will provide a list of any threats found, allowing the user to choose which files to delete or keep, or removing files that match.

Important steps to avoid viruses and other malware:

1. Delete spam and suspicious emails without opening or activating any attached files or links; it is also important not to open, forward, or reply to suspicious email. The user should be suspicious if:
    a. An attachment or link in an email message is unexpected or unsolicited
    b. The email is not addressed to you by name
    c. You don't recognize the sender or the email says it is from a "friend"
    d. You can't determine why the file or link was sent to you
2. Do not send, forward or open electronic greeting cards, animations, games, joke programs, chain letters, screen savers, songs, videos or images. In addition, they can needlessly consume system resources.
3. Ensure that all current patches/updates are installed for your computer's operating system and applications.

4. Do not download or install unknown software or software from an unknown source. Even if it is "free", you may get more than you realized (e.g., spyware, adware, etc).
5. Back up your important data and mobile devices to separate media, such as a CD/DVD, an online backup service, flash drive, or a server. Store backups in a safe place.

Today, many users install antivirus software that can detect and eliminate the malware from their local computers. An antivirus can be effective in many circumstances, but can also miss very new or particularly sophisticated malware, and is not a complete solution to combatting malware. Still, it is a commonly employed mechanism to protect against malware infections, and for an antivirus to be effective, it’s necessary to be able to quickly identify malware behaviors and generate appropriate signatures or behavioral profiles.

How does Antivirus software work?

There are two common methods that an antivirus software application detects viruses –

- Signature Detection
- Behavior Detection

Signature Detection is the most common technique used by an antivirus software. This involves searching for known malware signatures that are essentially strings of bits that are unique to a particular type of malware. But, it has a main disadvantage that it only protects against malware for which signatures have been updated in the antivirus service's database and not against previously unknown malware ("zero day attacks").

Behavior detection uses heuristic algorithms to detect malicious behaviors. Behaviors are essentially a set of actions (operations) on objects (system resources). The algorithms analyze the patterns to identify potential threats. This method has the ability to detect new viruses for which anti-virus security firms have yet to define a "signature," but it also gives rise to more false positives than using signatures.

Chapter 3

Related Work

3.1 Approach

This thesis evaluates different kinds of malware and their behavior that are present in a huge malware collection. Initially, the malware are encrypted on the storage device, so they have to be decrypted by providing appropriate passwords. Most of the malware are Windows executables, with an .EXE extension. Analyzing the malware directly on physical hardware is dangerous because it can easily cause negative effects while it runs and damage the OS, applications, and data. In order to analyze the behavior of the malware samples, we used third-party tools and some custom tools written in python. By using this information, we get to know the malware's effects in detail. All the information that's collected is saved into a database, so that characteristics of malware can be easily queried and correlated.

3.2 Previous Research Methods

A lot of research has been done in the subject of IT security and malware incidents. Some information is taken from technical papers and journals with credible sources.

In all the technical papers, the researchers have classified malware by implementing different techniques for analysis. Some of them are described below:

- To extract malware behavior they observed all the system function calls performed in a virtualized execution environment and shown how the accuracy of the classification process can be improved using a phylogenetic tree. The phylogenetic trees were assessed using known antivirus results so that only a few malware behaviors were wrongly classified (Ref. 4).
- The innovative collaborative architecture for malware analysis aims for early detection and timely deployment of countermeasures. These nodes send alerts to intermediate managers that, in their turn, communicate with one logical collector

and analyzer. Relevant information is determined by the automatic analysis of the malware behavior in a sandbox, and countermeasures are then sent to all the cooperating networks. Cyphered communications among components help prevent the leakage of sensitive information and allow the pairwise authentication of the nodes involved in the information sharing. This approach is only used for network communication analysis (Ref. 5).

- By using different honeypot technologies to increase the variety of malware collected, they presented a daemon tool developed to grab malware distributed through spam and a pre-classification technique that uses antivirus technology to separate malware into generic classes. It is a minimal approach so that there is not much analysis information about the malware (Ref. 6).
- Based on function calls and control flow analysis, according to the identification of suspicious behavior, the technique implements a strategy of detection from malicious binary executables (Ref. 7).
- Classification of malware behavior proposes three stages: (a) behavior of collected malware is monitored in a sandbox environment, (b) based on a corpus of malware labeled by an anti-virus scanner a malware behavior classifier is trained using learning techniques, and (c) discriminative features of the behavior models are ranked for explanation of classification decisions. By using honeypots they detected novel instances of malware (Ref. 8).
- They proposed a framework for the automatic analysis of malware behavior using machine learning. It allows for automatically identifying novel classes of malware with similar behavior (clustering) and assigning unknown malware to these discovered classes (classification). Based on both, clustering and classification, they propose an incremental approach for behavior-based analysis, capable of processing the behavior of thousands of malware binaries on a daily basis. The incremental analysis significantly reduces the run-time overhead of current

analysis methods, while providing accurate discovery and discrimination of novel malware variants (Ref. 9).

- Signature-based detection is the most broadly used commercial antivirus method; however, it fails to detect new and previously unseen malware. Supervised machine-learning models have been proposed in order to solve this issue, but the usefulness of supervised learning is far from perfect because it requires a significant amount of malicious code and benign software to be identified and labelled beforehand. One paper proposes a new method to detect unknown malware. Collective classification is a type of semi-supervised learning that presents an interesting method for optimizing the classification of partially-labelled data. In this way, collective classification algorithms build different machine-learning classifiers using a set of labelled (as malware and legitimate software) and unlabeled instances and performed an empirical validation demonstrating that the labelling efforts are lower than when supervised learning is used, while maintaining high accuracy rates (Ref. 10).
- In other related work, the researchers proposed a novel algorithm for constructing a control flow graph signature using the decompilation technique of structuring. Similarity between structured graphs can be quickly determined using string edit distances. To reverse the code packing transformation, a fast application level emulator is proposed. To demonstrate the effectiveness of the automated unpacking and flow graph based classification, they implement a complete system and evaluate it using synthetic and real malware. The evaluation shows their system is highly effective in terms of accuracy in revealing all the hidden code, execution time for unpacking, and accuracy in classification (Ref. 11).
- In other related work, the researchers proposed a behavior-based automated classification method. Depending on behavioral analysis, they characterize a malware behavioral profile in a trace report. This report contains the status change caused by the executable and event which are transferred from corresponding Win32 API calls and their certain parameters; and extract behavior unit strings as

features which reflect the behavioral patterns of different malware families. They use string similarity and information gain to reduce the dimension of feature space. Comparative experiments with a real world data set of malicious executables show that their proposed method can classify malware into different malware families with higher accuracy and efficiency (Ref. 12).

All the above research experiments show only the classification of malware in different approaches like APIs, function calls, signature based detection, honeypots but they have only limited analysis information and some drawbacks too. My application is more efficient and helps to catalog and classify large collections of unknown malware using behavior analysis.

3.3 Research Work

This project requires a dedicated machine, in order to easily perform the analysis safely and, to move the malware database and associated analysis tools easily, all without endangering other computer systems.

I run the virtualization software such as VMWare Workstation in my local system and created both Windows and UNIX OS VMs to work on the malware (.exe) files, to extract its information using software tools, as well as some custom programming in Python and Java.

Initially, I worked on the signature API's to find the MD5, SHA1, SHA256 hashes of malware samples using VirusTotal and JSON scripts. This is a free online service that analyzes files and URLs enabling the identification of viruses, worms, trojans and other kinds of malicious content detected by antivirus engines and website scanners. At the same time, it may be used to detect false positives, i.e. innocuous resources detected as malicious by one or more scanners.
By using this information, we can find the binaries, libraries, dll dependencies, and virus name of the malware.

3.4 Anubis

Anubis is a tool for analyzing the behavior of Windows PE-executable (binaries) files. Execution of Anubis results in the generation of a report file in HTML, XML, text, and PDF formats that contains very detailed information about the analyzed binary. The analysis is based on running the binary in an emulated environment and monitoring its execution. The analysis focuses on the security-relevant aspects of a program's actions, which makes the analysis process easier. Because the domain is more fine-grained it allows for more precise results. It is an ideal tool to obtain a quick understanding of the purpose of an unknown binary.

- The generated report includes detailed data about modifications made to the Windows registry or the file system, about interactions with the Windows Service Manager or other processes and of course it logs all generated network traffic.

Limitations:

- The Anubis has the limitations to the users. The users cannot submit the malware files in a large amount at a time. Also the users can submit only few hundred files in a day for the analysis. Ref: http://anubis.iseclab.org/?action=home

3.5 ClamAV

Clam AntiVirus is an open source (GPL) anti-virus toolkit for UNIX, designed especially for e-mail scanning or mail gateways. It provides a number of utilities including a flexible and scalable multi-threaded daemon, a command line scanner and advanced tools for automatic database updates. This allows for good offline protection, advanced archive and unpacking support, and custom signature creation for customizable security. ClamAV utilizes advanced cloud-based, community-based, and integrated ClamAV detection technology to help secure our PC. The newest release fully integrates the core ClamAV detection engine to provide exceptional offline protections against the latest malware threats. (Ref. 13)

Features:

- Real-time detection
- Scheduled scanning
- Intelligent Scanning – Fast and configurable smart scans
- Custom Detection – Using de facto standard ClamAV signature language
- Quarantine
- Sign UI

There are other packages like clamd, clamdscan, libclamAV, freshclam and binaries which are used for configuration, executables, archives and compressed files, etc.,

3.5.1 clamd

Before we start using the daemon we have to edit the configuration file (in case clamd won't run):

```
$ clamd ERROR: Please edit the example config file /etc/clamd.conf.
```

This shows the location of the default configuration file.

<u>Setup Auto-update:</u>

3.5.2 freshclam

freshclam is the automatic database update tool for Clam AntiVirus. It can work in two modes:

- interactive - on demand from command line
- daemon - silently in the background

freshclam is advanced tool: It supports scripted updates (instead of transferring the whole CVD file at each update it only transfers the differences between the latest and the current database via a special script), database version checks through DNS, proxy servers (with authentication), digital signatures and various error scenarios.

Quick test:

run freshclam (as superuser) with no parameters and check the output. If everything is OK you may create the log file in /var/log.

Now we should edit the configuration file freshclam.conf and point the UpdateLogFile directive to the log file. Finally, to run freshclam in the daemon mode, execute:

```
# freshclam -d
```

3.5.3 Clam daemon

clamd is a multi-threaded daemon that uses libclamav to scan files for viruses. It may work in one or both modes listening on:

- Unix (local) socket
- TCP socket - The daemon is fully configurable via the clamd.conf file.

3.5.4 Clamdscan

clamdscan is a simple clamd client. In many cases you can use it as a clamscan replacement. However you must remember that:

- It only depends on clamd
- Although it accepts the same command line options as clamscan most of them are ignored because they must be enabled directly in clamd, i.e. clamd.conf

- In TCP mode scanned files must be accessible for clamd, if you enabled LocalSocket in clamd.conf then clamdscan will try to work around this limitation.

3.5.5 Clamscan

clamscan writes all regular program messages to stdout and errors/warnings to stderr. You can use the option --stdout to redirect all program messages to stdout. Warnings and error messages from libclamav are always printed to stderr. A typical output from clamscan looks like this:

```
/tmp/test/removal-tool.exe: Worm.Sober FOUND
        /tmp/test/md5.o: OK
        /tmp/test/message.c: OK
        /tmp/test/error.hta: VBS.Inor.D FOUND
```

When a virus is found its name is printed between the filename: and FOUND strings. In case of archives the scanner depends on libclamav and only prints the first virus found within an archive:

```
dkura@localhost:/tmp$ clamscan malware.zip
 malware.zip: Worm.Mydoom.U FOUND
```

3.5.6 libclamAV

Libclamav provides an easy and effective way to add a virus protection into our software. The library is thread-safe and transparently recognizes and scans within archives, mail files, MS Office document files, executables and other special formats.

3.6 7-zip

7-Zip is open source software.

Features:

- Supported formats:
    - ✓ Packing / unpacking: 7z, XZ, BZIP2, GZIP, TAR, ZIP and WIM
- For ZIP and GZIP formats, 7-Zip provides a compression ratio that is 2-10 % better than the ratio provided by PKZip and WinZip
- Strong AES-256 encryption in 7z and ZIP formats
- Self-extracting capability for 7z format
- Integration with Windows Shell

This 7-zip software plays an important role to decrypt the Windows executable malware files and from password protection, too. It gives faster performance with results.

In this thesis project, my first step is to unzip the password protected malware files. Later these files are analyzed by submitting them to Anubis. Here, I used Martin's code: Reading password protected Zip files in Java (Ref. 14). He implemented the class and used the ZIP File Format Specification as the source of information. He used the 7-zip project (C++) as a reference during the debugging to verify his understanding of the ZIP spec. and the CRC algorithm.

```
static {
    for (int i = 0; i < 256; i++) {
        int r = i;
        for (int j = 0; j < 8; j++) {
            if ((r & 1) == 1) {
                r = (r >>> 1) ^ 0xedb88320;
            } else {
                r >>>= 1;
            }
        }
```

```
            CRC_TABLE[i] = r;
        }
    }
```

These are the limitations:

- Only the “Traditional PKWARE Encryption” is supported
- Files that have the “compressed length” information at the end of the data section (rather than at the beginning) are not supported

Chapter 4

Software Development – Desktop Application

4.1 Python

Python is a remarkably powerful dynamic programming language that is used in a wide variety of application domains. It lets us work more quickly and integrate our systems more effectively.

Key Features:

- very clear, readable syntax
- strong introspection capabilities
- intuitive object orientation
- natural expression of procedural code
- full modularity, supporting hierarchical packages
- exception-based error handling
- very high level dynamic data types
- extensive standard libraries and third party modules for virtually every task
- embeddable within applications as a scripting interface

4.1.1 Calling Python from Java

In this thesis, I have called Python from the command-line by using the Python interpreter. I used the process to run the Python methods in the Java program. The following sample code shows the process:

```
Process p = Runtime.getRuntime().exec();

PythonInterpreter.initialize(System.getProperties(),  newString[0]);

PythonInterpreter interp = new PythonInterpreter();

Interp.execfile("example.py");
```

With this approach, we have better control and debugging capabilities.

4.1.2 Use of Python in project

Python is used for two reasons:

- Anubis: malware analysis tool script called 'submit_to_anubis.py' was developed in Python.
- 7-zip: After the program execution starts, 7-Zip calls the 'run.py' method which gives the filepath to the malware files through the socket.

4.2 ClamAV Installation Setup and Steps

Download and install the 'clamAV' and 'pyclamd' setup file. 'Start_clamd.bat' file was written to update the location path for the clamd configuration file.

- C:\"Program Files"\clamAV64\Setup-x64\clamd –config file=C:\Users\Public\clamd\clamd.conf

```
>>>pyclamd = Extension('pyclamd', sources = ['pyclamd.py'])
        # Install : Python setup.py install
        # Register : Python setup.py register
```

4.2.1 ClamAV Integration

'run.py' has the file path program of a malware directory path present in an external hard drive.

The next step is to create the log files in a log folder and edit the configuration files for both the clamd and freshclam properties. Freshclam is mainly used for updating the Clam Antivirus. It creates the database folder with the four index files which are used for 'Virus Signature' written in ClamAV. They are:

- bytecode.cvd

- daily.cvd
- main.cvd
- mirrors.dat

4.3 pyClamd

‘pyClamd is a Python interface to Clamd (Clamav daemon). By using pyClamd, you can add virus detection capabilities to your Python software in an efficient and easy way’. ‘setup.py’ is used for the pyclamd installation which helps in providing the libraries for ClamAV. ClamAV runs automatically in MalwareDetector.jar file. It extracts the virus name or error string in the return value.

4.4 Work Procedure

Our malware categorization application is a desktop application that runs with the updated Java ‘7’ version in our local system. Initially, the program execution starts with the ‘run’ batch file. It will open a JFrame popup window. I used MySql database for the backend to save the analysis report data. For the first time, we have to set the configuration properties by selecting ‘Load from Config’ button for the successful connection establishment with the database. Then ‘Create Schema’ should choose to create the tables with empty columns and data (Fig. 2).
Browse the directory path to the malware location which has to scan the windows executable (.exe) files for the analysis by selecting the ‘Browse’ button. By clicking the ‘Scan’ button, the directory will be scanned for the malware and extracts it by decryption which displays in the (Fig. 3) and submitted to the Anubis for analysis.

All malware are present in an external hard drive with the encrypted password protection in a zipped file format. All filenames are named with the same name called ‘malware’ with the same password ‘infected’. Using both the 7-zip software installation and the Martin’s code: Reading encrypted zipped files in Java, the malware files are decrypted whether it is password protected or not, and unzipped by scanning the directory path.

4.5 Extracting malware file from zip file with password protection

After running the application successful by submitting malware files, it has the capacity to take about fourteen minutes for one thousand files by decrypting and extracting the malware from the zip files. It runs faster, secure, and no data loss.

4.6 Limitation for submission of malware files to Anubis for analysis:

- We cannot exceed the limitation of submitting files to Anubis of not more than a thousand in a day.
- We should submit only in multiple batches with a small number of files in it. This application submits only 15 files in a batch for every thirty minutes. It means a total of 700 files are submitted in a day by itself with the help of a batching algorithm. These batches will get the analysis done and return with its respective URL report links.

4.7 Scanning for .EXE files which haves less than 8MB in size

According to the Anubis tool, it allows only less than 8 MB file size for analysis. In this thesis, I used the 'listing algorithm' where the condition is that we can measure the file size of each malware. This setup is written in our configuration file. While scanning, if the malware file size has more than 8 MB then it is skipped and continues the execution process to find the next malware file to submit the Anubis for analysis.

The results of a directory are submitted to a third-party Anubis (malware analysis tool) directly for the behavior analysis of a malware. It generates results in a URL report formats like HTML, XML, text & PDF for each malware. Each format has the same results. I used the XML parsing for saving the data into the database. According to the data the columns will be created 'dynamically'. The data will insert according to the column names generated in the XML report. In this way, no data will be skipped and the execution is done without any exception.

4.8 Fetching XML version of an HTML report from the Anubis URL

Using an XML parser the data from Anubis is fetched and saved into the database.

XML parsers are of two types:

1) **SAX parser** - parse the whole document at a time. It has the interface and XML handler classes. I used this parser in my project because it is a smart way of implementing and finding the scope of the document with the structured data.
2) **DOM parser** – Mainly used for unstructured data.

Chapter 5

Database Modeling

Initially, I established a successful database connection in between the Java program and the MySql database for saving the XML parsed data into the database. In the coding implementation, I made JDBC connectivity with an Oracle thin driver. The connection string is loaded in a configuration file which can be easily edited according to the user requirements for the successful connection establishment. This configuration file has the Connection name, Port no., Schema name, Username, Password, 7-Zip file path, and zipPassword (Fig. 2).

5.1 Create and Insert tables

A script was created in .sql and code written for automatic creation and insertion tables in the Java application program. The database is developed with the fixed 13 tables with only few known fixed columns. But, as per the XML parsing discussion, columns are added dynamically into the database as needed, with column names based on the XML tags from the Anubis Server.

5.2 Tables Structure

'Malware_file_info' acts as a master table in the database. The columns are Serial No., File Id, Virus name and Date Timestamp. Serial No. is the unique number generator for each malware file. File Id which maps to other tables depending on xml structure and to save the related dependency data. It analyzes different structure of malware to incorporate into database.

5.3 Data Comparison

In the database, we can query and compare the data in between any malware or by any attributes. This is the best way, to analyze the malware behavior without affecting any computer and we can take the necessary precautions by knowing the threat level of a malware and try to resolve the issues.

The following thirteen (13) tables are created with the database schema ‘Scanner’:

- Malware_file_info
- Report_Version
- Configuration
- Analysis_Subject
- Popups
- Global_Network_Activities
- Global_file
- Dll_dependencies
- Registry_Activities
- File_Activities
- Process_Activities
- Network_Activities
- Popup

Evaluation and Results

I used a malware database provided to us by Danny Quest, which contains three million samples stored on an external USB hard drive. The application I developed scans the zipped files in a directory and extracts the malware files by decrypting if the files are password protected, then submitted to Anubis for analysis. For faster analysis, protecting the data, and minimizing the burden on the Anubis server, a batching algorithm is used, which submits one batch at a time, each containing 15 files. A delay of thirty minutes passes before the next batch is transmitted. In this way, 720 files are submitted in a day for analysis, which doesn't exceed the limitation imposed by the Anubis administrators. The retrieved reports from Anubis are temporarily saved in the application and the data from the reports is stored in the database.

The database columns are created dynamically. With the available results we can measure the threat level of a malware, categorize the malware families, how harmful when they reside on computers, and their effects.

The following tables show the results, that I have obtained after running the application for malware analysis:

No. of Malware files for submission	Time taken for the Extracting decrypted files from zip or Unzip folder in seconds	Time taken for submission to get Anubis URL report links in seconds
40	60	135
1,000	1240	3,375
99,138	11900	28,450

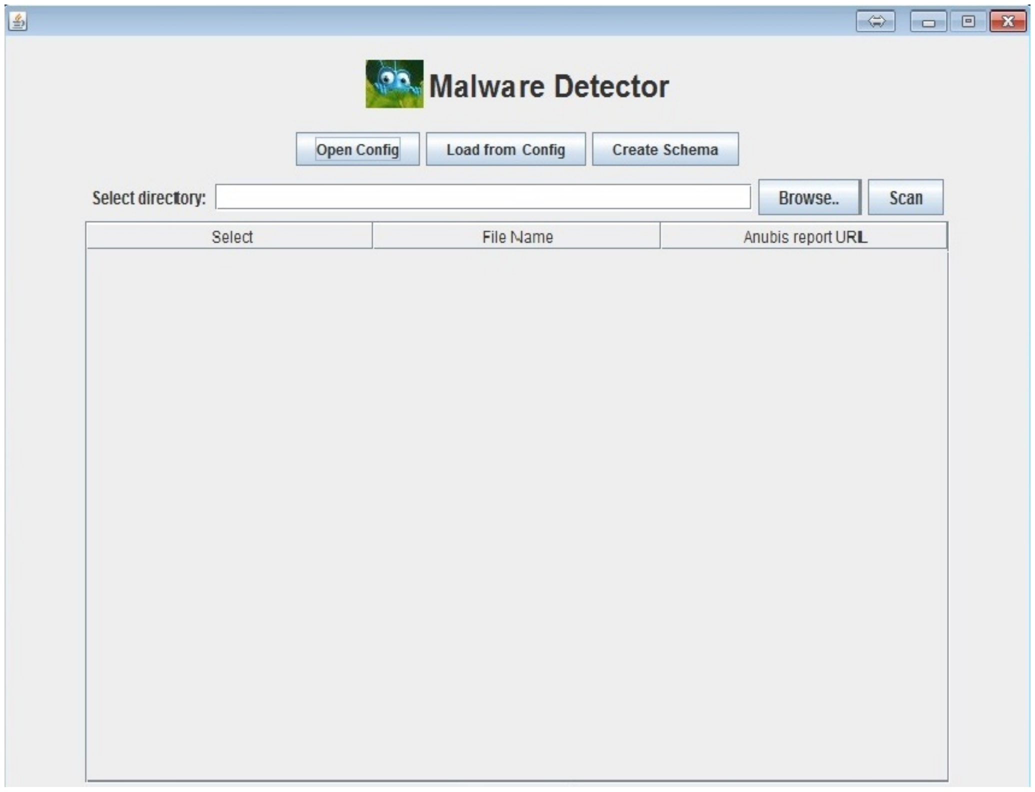
Table 1. Analysis URL reports without any Limitations in submission of file to Anubis Sandbox server.

No. of Malware files for submission	Time taken for the Extracting decrypted files from zip or Unzip folder in seconds	Time taken for submission to get Anubis URL report links in seconds
15	27	1800
15	25	1800
15	30	1800
……....	**………**	**………**
……...	**………**	**………**

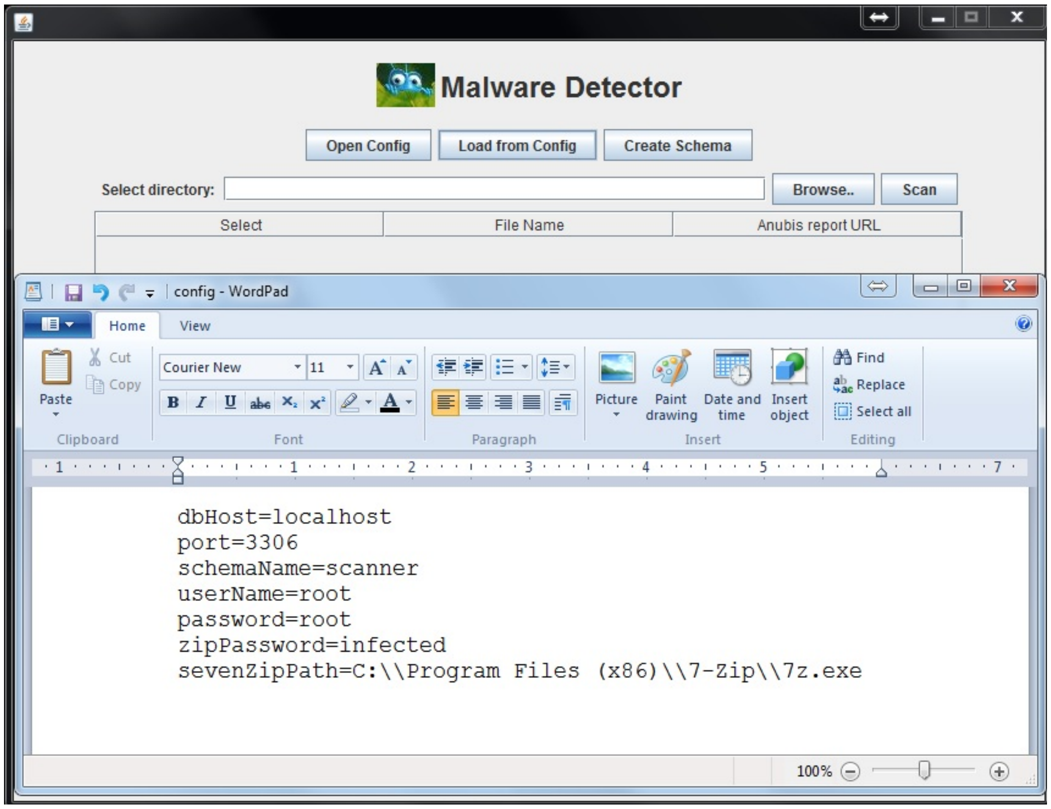
Table 2. Malware submissions in batches. Anubis URL reports with Limitations

Total = 720 malware files submission for 24 hours (1 day) to Anubis for analysis

The following screenshots shows the step-by-step procedure:

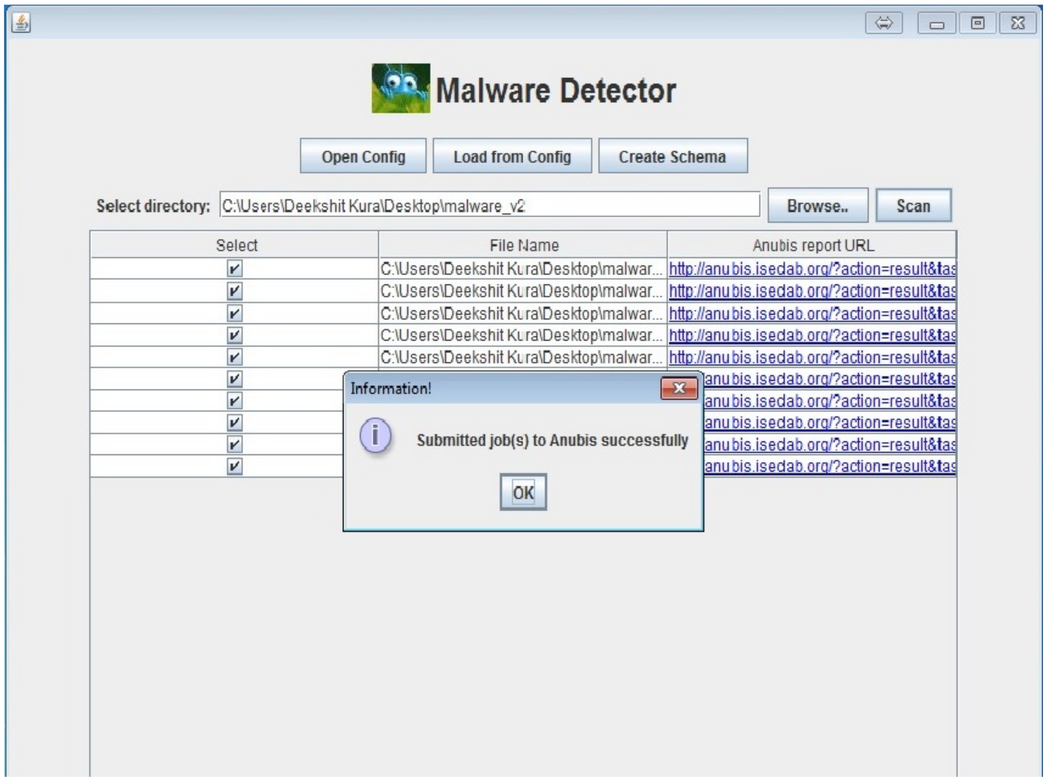


Fig(1) Malware Detector - Application window



```
dbHost=localhost
port=3306
schemaName=scanner
userName=root
password=root
zipPassword=infected
sevenZipPath=C:\\Program Files (x86)\\7-Zip\\7z.exe
```

Fig(2) Configuration file for the database connection

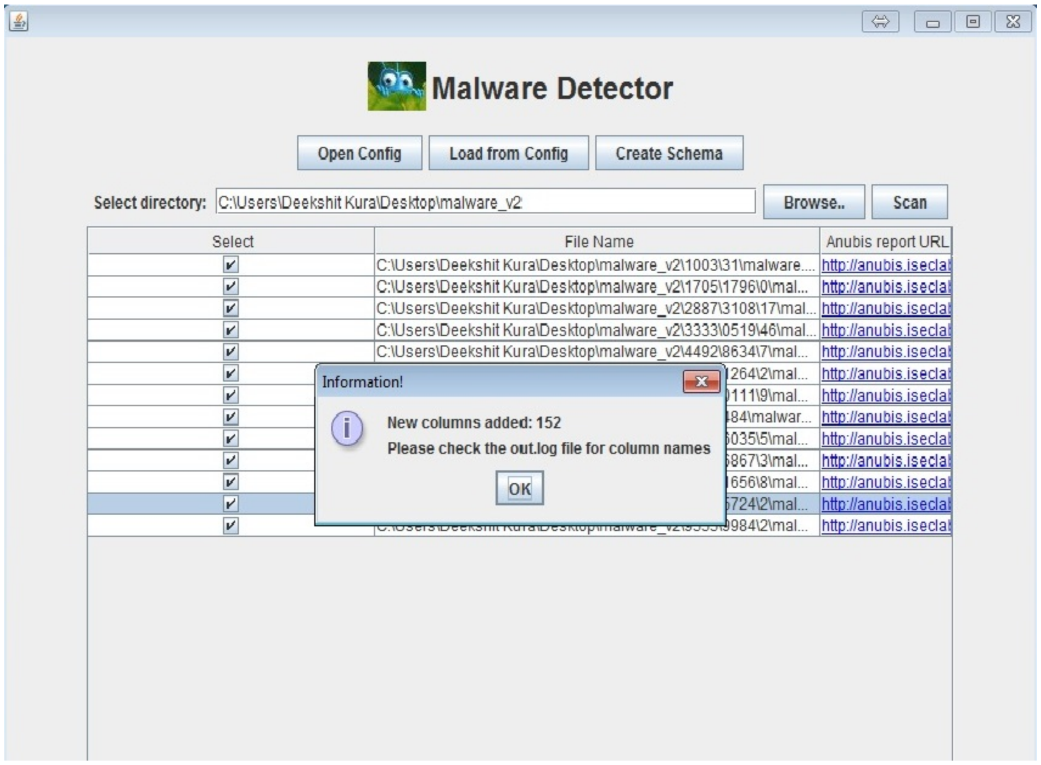


Select directory: C:\Users\Deekshit Kura\Desktop\malware_v2

Select	File Name	Anubis report URL
✓	C:\Users\Deekshit Kura\Desktop\malwar...	http://anubis.isedab.org/?action=result&tas
✓	C:\Users\Deekshit Kura\Desktop\malwar...	http://anubis.isedab.org/?action=result&tas
✓	C:\Users\Deekshit Kura\Desktop\malwar...	http://anubis.isedab.org/?action=result&tas
✓	C:\Users\Deekshit Kura\Desktop\malwar...	http://anubis.isedab.org/?action=result&tas
✓	C:\Users\Deekshit Kura\Desktop\malwar...	http://anubis.isedab.org/?action=result&tas
✓		anubis.isedab.org/?action=result&tas
✓		anubis.isedab.org/?action=result&tas
✓		anubis.isedab.org/?action=result&tas
✓		anubis.isedab.org/?action=result&tas
✓		anubis.isedab.org/?action=result&tas

Submitted job(s) to Anubis successfully

Fig(3) Bulk of malware are submitted to Anubis Sandbox for analysis



Select directory: C:\Users\Deekshit Kura\Desktop\malware_v2

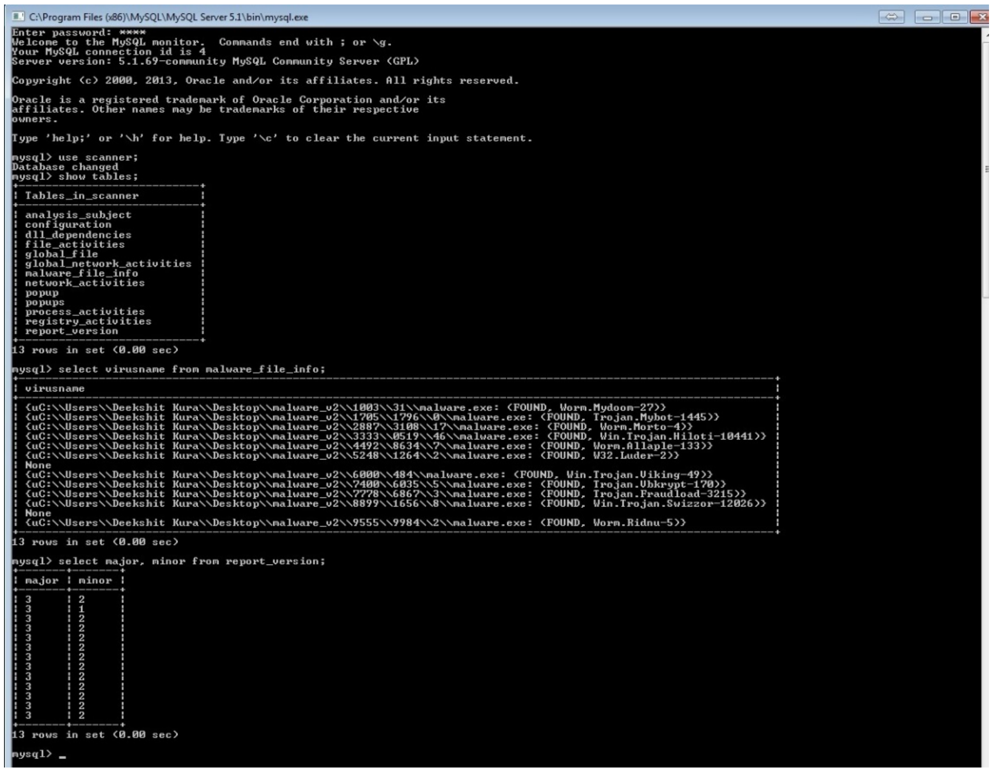
Select	File Name	Anubis report URL
✓	C:\Users\Deekshit Kura\Desktop\malware_v2\1003\31\malware....	http://anubis.iseclat
✓	C:\Users\Deekshit Kura\Desktop\malware_v2\1705\1796\0\mal...	http://anubis.iseclat
✓	C:\Users\Deekshit Kura\Desktop\malware_v2\2887\3108\17\mal...	http://anubis.iseclat
✓	C:\Users\Deekshit Kura\Desktop\malware_v2\3333\0519\46\mal...	http://anubis.iseclat
✓	C:\Users\Deekshit Kura\Desktop\malware_v2\4492\8634\7\mal...	http://anubis.iseclat
✓	1264\2\mal...	http://anubis.iseclat
✓	0111\9\mal...	http://anubis.iseclat
✓	84\malwar...	http://anubis.iseclat
✓	6035\5\mal...	http://anubis.iseclat
✓	6867\3\mal...	http://anubis.iseclat
✓	1656\8\mal...	http://anubis.iseclat
✓	5724\2\mal...	http://anubis.iseclat
✓	9984\2\mal...	http://anubis.iseclat

Information!

New columns added: 152
Please check the out.log file for column names

OK

Fig(4) XML parsing the results: New Columns are added into the database


C:\Program Files (x86)\MySQL\MySQL Server 5.1\bin\mysql.exe

```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.1.69-community MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use scanner;
Database changed
mysql> show tables;
+-----------------------------+
| Tables_in_scanner           |
+-----------------------------+
| analysis_subject            |
| configuration               |
| dll_dependencies            |
| file_activities             |
| global_file                 |
| global_network_activities   |
| malware_file_info           |
| network_activities          |
| popup                       |
| popups                      |
| process_activities          |
| registry_activities         |
| report_version              |
+-----------------------------+
13 rows in set (0.00 sec)

mysql> select virusname from malware_file_info;
+------------------------------------------------------------------------------------------------------------+
| virusname                                                                                                  |
+------------------------------------------------------------------------------------------------------------+
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\1003\\31\\malware.exe: (FOUND, Worm.Mydoom-27)}          |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\1705\\1796\\0\\malware.exe: (FOUND, Trojan.Mybot-1445)}  |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\2887\\3108\\17\\malware.exe: (FOUND, Worm.Morto-4)}      |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\3333\\0519\\46\\malware.exe: (FOUND, Win.Trojan.Hiloti-10441)} |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\4492\\8634\\7\\malware.exe: (FOUND, Worm.Allaple-133)}   |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\5248\\1264\\2\\malware.exe: (FOUND, W32.Luder-2)}        |
| None                                                                                                       |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\6000\\484\\malware.exe: (FOUND, Win.Trojan.Viking-49)}   |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\7400\\6035\\5\\malware.exe: (FOUND, Trojan.Vbkrypt-170)} |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\7778\\6867\\3\\malware.exe: (FOUND, Trojan.Fraudload-3215)} |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\8899\\1656\\8\\malware.exe: (FOUND, Win.Trojan.Swizzor-12026)} |
| None                                                                                                       |
| {uC:\\Users\\Deekshit Kura\\Desktop\\malware_v2\\9555\\9984\\2\\malware.exe: (FOUND, Worm.Ridnu-5)}       |
+------------------------------------------------------------------------------------------------------------+
13 rows in set (0.00 sec)

mysql> select major, minor from report_version;
+-------+-------+
| major | minor |
+-------+-------+
| 3     | 2     |
| 3     | 1     |
| 3     | 2     |
| 3     | 2     |
| 3     | 2     |
| 3     | 2     |
| 3     | 2     |
| 3     | 2     |
| 3     | 2     |
| 3     | 2     |
| 3     | 2     |
| 3     | 2     |
| 3     | 2     |
+-------+-------+
13 rows in set (0.00 sec)

mysql> _
```

Fig(5) MySql database: Querying analysis results of a malware to measure the threat level of a malware

Conclusion

We have successfully created a comprehensive malware behavior identification system that, classifies malware with the assistance of the custom application and results from the Anubis sandbox. The results of malware execution and identification are stored in a database, suitable for querying for particular malware functionalities and characteristics, and correlation between malware instances to identify common behavior.

This information is very helpful in identifying and classifying malware families. This application is fast, robust, and reliable for accurate results. With the available results we can measure the threat level of a malware, categorize the malware families, how harmful when they reside on host systems, and their effect. We can then take the appropriate precautions to avoid their effect on the host system.

Future Work

The work can be expanded by integrating with other third-party tools, similar to Anubis, to expand the detail of our malware behavioral analysis. Every tool has its own functionality and strengths and weaknesses, and may provide more detail on individual malware instances. We hope to find the tools that have no limitation in file size and increase in the capacity for submitting the number of malware files for analysis.

Bibliography

1) Malware Distribution: http://www.bullguard.com/bullguard-security-center/pc-security/computer-threats/malware-definition,-history-and-classification.aspx
2) Number and Types of malware in U.S: http://extremevoltages.blogspot.com/2009_09_01_archive.html
3) Refernce - http://en.wikipedia.org/wiki/Malware
4) J Comput Virol: Malware Behavioral Analysis, Springer-Verlag, France (2007)
5) Michele Colajanni; Daniele Gozzi; and Micro Marchetti: Collaborative architecture for malware detection and analysis, Springr-US, (2008)
6) Andre R. A. Gregio; Isabela L. Oliveria; Rafeal D. C. Santos; Andrio M. Cansian; Paulo L. de Geus: Malware distributed collection and preclassification system using honeypot technology, Orlando, USA (2009)
7) Cheng Wang; Zhengzhou; Jianmin Pang; Rongcai Zhao; Wen Fu: Malware Detection Based on Suspicious Malware Behavioral Detection, China (2009)
8) Konrad Rieck; Thorsten Holz; Carsten Willems; Patrick D'ussel and Pavel Laskov: Learning and Classification of Malware Behavior, (2008)
9) Konrad Rieck; Thorsten Holz; Carsten Willems; Philipp Trinius: Automatic Analysis of Malware Behavior using Machine Learning, (2011)
10) Igor Santos, Carlos Laorden and Pablo G. Bringas: Collective Classification for Unknown Malware Detection, Spain (2011)
11) Silvio Cesare; Yang Xiang: Classification of malware using structured control flow, Australia, (2010)
12) HengLi Zhao; HangZhou DianZi; Ming Xu; Ning Zheng; Jingjing Yao: Malicious Executables Classification Based on Behavioral Factor Analysis, Sanya (2010)
13) ClamAV: http://www.clamav.net/lang/en/

14) Martin. “Reading password-Protected zip files in Java” http://blog.alutam.com/2009/10/31/reading-password-protected-zip-files-in-java/ (2009)

<u>Other References:</u>

15) Jaime Devesa, Igor Santos, Xabier Cantero, Yoseba K. Penya and Pablo G. Bringas, S3Lab, Deusto Technological Foundation, Bilbao, Spain. “Automatic behaviour-based analysis and classification system for malware detection” (2010).
16) Michael Bailey, Jon Oberheide, Jon Andersen, Z. Morley Mao, Farnam Jahanian, Jose Nazario. “Automated Classification and Analysis of Internet Malware” (2007).
17) Danny Quist, Ph.D. Los Alamos National Laboratory. “VERA virtualization tool” (2011).

APPENDIX

Martin's Code: Reading file from password protected in Java:

```
public class Main {
   public static void main(String[] args) throws IOException {
      // password-protected zip file I need to read
      FileInputStream fis = new FileInputStream(args[0]);
      // wrap it in the decrypt stream
      ZipDecryptInputStream zdis = new ZipDecryptInputStream(fis, args[1]);
      // wrap the decrypt stream by the ZIP input stream
      ZipInputStream zis = new ZipInputStream(zdis);
      // read all the zip entries and save them as files
      ZipEntry ze;
      while ((ze = zis.getNextEntry()) != null) {
         FileOutputStream fos = new FileOutputStream(ze.getName());
         int b;
         while ((b = zis.read()) != -1) {
            fos.write(b);
         }
         fos.close();
         zis.closeEntry();
      }
      zis.close();
   }
}
```

run.py

```
>>>import sys

     import pyclamd

     filePath = sys.argv[1]

     cd = pyclamd.ClamdNetworkSocket()

     ret = cd.scan_file(filePath)

     print ret
```

Setup.py

‘setup.py’ is the installation file for ‘pyclamd daemon’.

```
>>>import pyclamd

        setup (name = 'pyClamd',

        version = pyclamd.__version__,

        package_dir={'pyclamd': ''},

        packages=['pyclamd'],

        author = 'Alexandre Norman',

        author_email = 'norman()xael.org',

        license ='LGPL',

        keywords="Python, clamav, antivirus, scanner, virus, libclamav",

        url = 'http://xael.org/norman/Python/pyclamd/',

        include_dirs = ['/usr/local/include'],
```

Vita

Deekshit Kura was born in India and received his Bachelor's degree in Information Technology –from the Jawaharlal Nehru Technology University of Hyderabad(JNTUH). His undergraduate final year project is on 'Minimizing Delay and Maximizing Lifetime of Wireless Sensor Networks with AnyCast' is to reduce the event-reporting delay and to prolong the lifetime of wireless sensor networks. He was admitted to the graduate school of the University of New Orleans in Fall 2011, where he worked under the guidance of Professor Golden Richard III and built a malicious software behavior analysis and its database.

Through part of his studies, he worked as a graduate assistant in UNO as a .Net Web Application Developer. He also interned at the Mirliton Media as a .Net Developer for two months in the Summer 2012. His graduate studies were concentrated on distributed systems, concurrent programming, and database management systems.