Malware Meta Crawler for MASS

MA-INF 3309 - Malware Analysis
Lab Report
Winter Semester 2016.17
University of Bonn

Ehab Qadah

May 7, 2017

Table of Contents

Abstract. On a daily basis, new malware samples are discovered. This makes the software vulnerabilities analysis one of the top concerns for organizations. The automatic identification of vulnerable software inside the organization is fundamental to avoid cyber-attacks. In this paper, we discuss two techniques to automatically monitor software vulnerabilities using open standards and public vulnerability information repositories, and alternative method to identify a vulnerable software using information obtained from social media platforms.

1 Introduction

In last decade, the usage of the Internet has increased and adopted all sectors of business and industry as result of the digital revolution. On the other hand, the wide usage of Internet creates a new opportunities for Cyber criminals to perform their malicious activities such as information theft and espionage. Malicious Software (malware) is any software has a harmful intention and abuse the user's computer [1]. Malware is a common tool to perform cyber attacks that can be in different forms such as worm, virus, Trojan and spyware [2]. According to Symantec, in 2015, 431 million of new malwares were discovered [3], which means over one million per day. To protect the Internet's users the malware researchers community try hardly to study these malwares, in order to build the counter measures and detect the new malware software or their malicious behavior, using different malware analysis techniques like static or dynamic analysis of malware samples [4].

In this work, we aim to develop malware meta crawler to feed the Malware Analysis and Storage System (MASS) [5] server database with new malware samples retrieved from on-line sources and repositories that provide malicious domains, Uniform Resource Identifiers (URIs) [1] and binaries, the retrieved samples are known to be connected with malware activities or deliver malicious payload. Furthermore, the malware meta crawler contains analysis units that analyze the malware samples to enrich the samples with additional related data (e.g., IP of a malware domain). Furthermore, it detects the relations between the malware samples and submit it to the MASS server.

[1] `https://tools.ietf.org/html/rfc3986`

To sum up, the goal of our system is to build a comprehensive database of malicious softwares, to make the malware samples continuously available in one place, which helps the security researchers.

The remainder of this report is organized as follows. In Section 2, we present the related work and fundamental background . Section 3 presents the general system overview. In Section 4 we give the implementation details. Section 5 provides the evaluation results. And finally, Section 6 gives the overall conclusion and future work.

2 Background and Related Work

In this section, we review some of the related work to our system and the required concepts that are used throughout our work. First, we provide an overview of the MASS echo system, malware types and analysis, related and similar systems and the reference on-line malware samples repositories. Finally, we give a brief background of Python.

2.1 Malware Analysis and Storage System (MASS)

The Malware Analysis and Storage System (MASS) serves as a platform for providing malware samples and analysis results [5]. All collected malware samples and analysis results (reports) are stored in a database on the MASS server. The MASS database contains malware samples submitted by malware researchers or retrieved by the malware meta crawler component. The MASS server is connected to several analysis systems which ease the process of sample reception and analysis. In addition to that MASS provide a web interface and REST APIs to access the samples and analysis results information.

The aim of MASS software is to provide the malware researchers a collaboration platform for malware analysis.

2.2 MASS API Client

The MASS API Client project provides a REST API interface to the MASS server operations [6], it is currently under the development phase. Nevertheless, we utilize the client's functionalities to submit the malware samples to the MASS server and develop analysis systems to enrich the information of the submitted malware samples.

2.3 Categories of Malware

This section provides an overview of the different types and classes of the malware samples. The malwares are categorized based on their similar characteristic and behavior, the following are some of the malware types were observed crossed the globe [7]:

- **Bot:** is a malware program utilizes the exploited system to perform malicious activities such as Spaming and involving in Denial of Service (DoS) attacks [8]. Usually the bot is remotely connected to Command & Control (C&C) server (i.e., botmaster) to receive the instructions of new malicious activities and updates.
- **Spyware:** is a malicious code collects sensistive information without the knowledge and permission of the system's user [7].
- **Virus:** is a malware program that requires to be attached to other host program in order to execute [9].
- **Worm:** "is a program that can run independently and can propagate a fully working version of itself to other machines"[9].

2.4 Malware Analysis

This section provides an overview of the malware analysis techniques (i.e., static and dynamic analysis). Bayer, Ulrich, et al. [1] define the malware analysis as the process of identifying and understanding the capabilities and goals of a malicious software sample.

2.4.1 Static Analysis

The static analysis is a technique to analysis the malware sample by generating the corresponding assembly code to understand the control and data flow of the sample [1]. This process does not require to execute the malware executable.

2.4.2 Dynamic Analysis

While the dynamic analysis approach is to study the behavior of malware sample by executing it in isolated environment [1]. This process requires to run the malware executable on a certain environment and find the effects of execution on the host system.

2.5 Related Systems

In this section, we provide a brief overview of similar tools to our system, namely, Maltrieve and Ragpicker.

2.5.1 Maltrieve

Maltrieve is an open source command line tool that retrieves malware samples from their sources [10], it fetches the malware URLs from different sites to download them and upload the samples to different stores such as VxCage [2]. On the other hand, Maltrieve does not provide any kind of analysis processing on the malware samples.

2.5.2 Ragpicker

Ragpicker is python based malware crawler that provide analysis and report functionalities [11]. It fetches malware URLs from different on-line sites, and provides processing functionalities (e.g., anti-virus scan, checking for suspicious checksum and check the IP for reputation) and reporting options like saving the analysis result in JSON format or save the samples in VxCage repository.

2.6 Sources of Malware Samples

This section present the online sources are used in the malware meta crawler to retrieve the malware samples, while more sources can be supported in future. The following are the supported sources that contain different malware samples information:

- Malware Domain Blocklist[3]: that provides a list of malware domains that are know to used for malicious purposes.
- Malc0de [4]: a database of URIs that contain malicious executables and binaries.
- ZeuS Tracker [5]: that provides RSS feeds of IPs and domains of malicious ZeuS hosts.

[2] https://github.com/botherder/vxcage
[3] http://www.malwaredomains.com/
[4] http://malc0de.com/database/
[5] https://zeustracker.abuse.ch/feeds.php

2.7 Python

The Python programming language [12] is a high-level, easy to use, general, purpose language, which supports object-oriented and functional programming programing paradigms. It is one of most popular languages in the scientific research. For those characteristics and others we have selected it to develop our system (i.e., Malware Meta Crawler for MASS).

3 System Overview

- idea +problem - formal algorithmic description - process flow diagram or any sort of charts

This section outlines the architecture and main building blocks of our system (Malware Meta Crawler for MASS).Furthermore, the process flow of the system is described. With its current functionality, our is a tool for retrieving the malware samples from different on-line channels to feed the malware database of the MASS server, also it contains analysis systems to automatically receive the submitted new samples, in order to add more information about the samples (e.g., find IP of malicious domain) and build the relation between the related malware samples (e.g., connect the malware executable file to its URI).

3.1 Architecture Overview

Figure 1 shows the general architecture of the Malware Meta Crawler for MASS. The system is divided into two principal subsystems, related to samples collection and processing respectively.

Samples collection: This subsystem is responsible for the retrieving of malware samples from the different on-line repositories were described the in Section 2.6. Then the fetched samples are mapped and parsed to the corresponding type (i.e., Domain, URI and File) by this subsystem. Furthermore, this subsystem submits the retrieved malware samples to the MASS server using the MASS API client interface.

Samples processing: This subsystem enriches the information of the malware samples retrieved by the sample collection subsystem, also it constructs the relation between the related malware samples. The following are the main modules (i.e., analysis units) of the sample processing subsystem, which automatically receive the new samples submitted to the MASS server:

- **URI samples Analysis Client:** this module is responsible to pull the new malware samples of URI type from the MASS server, in order to find the domain of each URI and submit the relation between them to the MASS server.

- **File Samples Analysis Client:** this module identify the URI of file samples using predefined regular expression, then it try to download the file from its source and submit it to the MASS server. In addition, it generates the

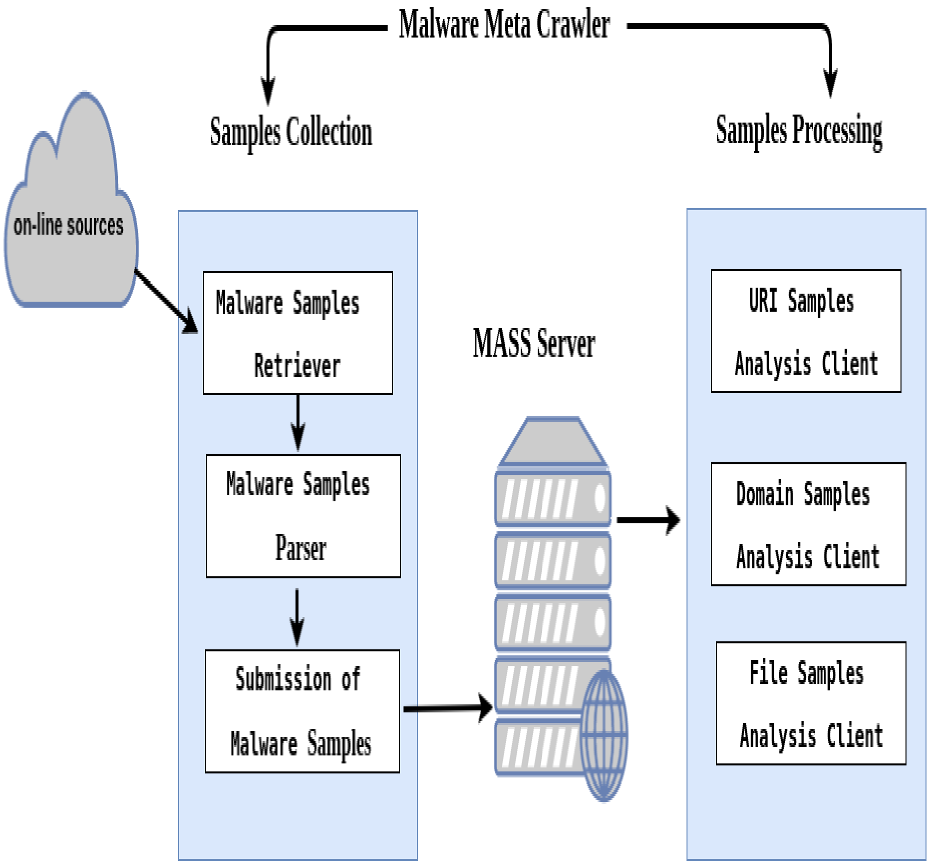
Malware Meta Crawler
Samples Collection
Samples Processing
on-line sources
Malware Samples Retriever
Malware Samples Parser
Submission of Malware Samples
MASS Server
URI Samples Analysis Client
Domain Samples Analysis Client
File Samples Analysis Client

Fig. 1. Generic architecture of the Malware Meta Crawler for MASS System.

relation between the sample file and origin URI sample.

- **Domain samples Analysis Client:** this module receive all new domain malware samples, then, it looks up for the domain's IP and connect them by the submitting the IP sample and the relation between them to the MASS server.

3.2 Process Flow of System

This section presents the process flow of the Malware Meta Crawler system that retrieves the malware samples from different on-line sources to submit them to the MASS server database. Figure 2 illustrates the internal process of the proposed system.

The process begins by collecting the malware samples from the mentioned sources in Section 2.6, and these samples include malicious domains and URIs for for infected websites or malware files. The system checks the retrieved samples to filter the old samples were processed and submitted before, and cache information about the new malware samples in internal storage for later checking. Then the system submits all new malware samples to the MASS server using the offered REST APIs.

This process is running again after waiting for the next trigger, which could be automate the execution using cron job that runs the system after certain period of time or even manual re-execution. This work flow will continue so that the system can continuously retrieve the new on-line malware samples.

On the other hand, the MASS schedules new analysis requests for the summited new malware samples by the first part of the system's work flow. Then the three analysis clients pull the scheduled analysis requests that contain the information of new malware samples to enrich their information and generate the relation between the related samples. The following are the analysis processes:

- **Enrich information of URI samples**: this process receives all new URI samples to find the URI's domain and submit the relation between the two samples to MASS.
- **Enrich information of domain samples**: this process receives all new domain samples in order to look up for the domain's IP and create a relation between them, received domains includes domains submitted by the system from the on-line repositories and the computed domains from previous process (i.e., domains of URI samples).
- **Enrich information of URI file samples**: this process receives all new URI samples to filter the URIs that belong to malware files, using a configured regular expression, then the corresponding file is downloaded and pushed to the MASS server. In addition, this process submits a relation between the downloaded file and its original URI.

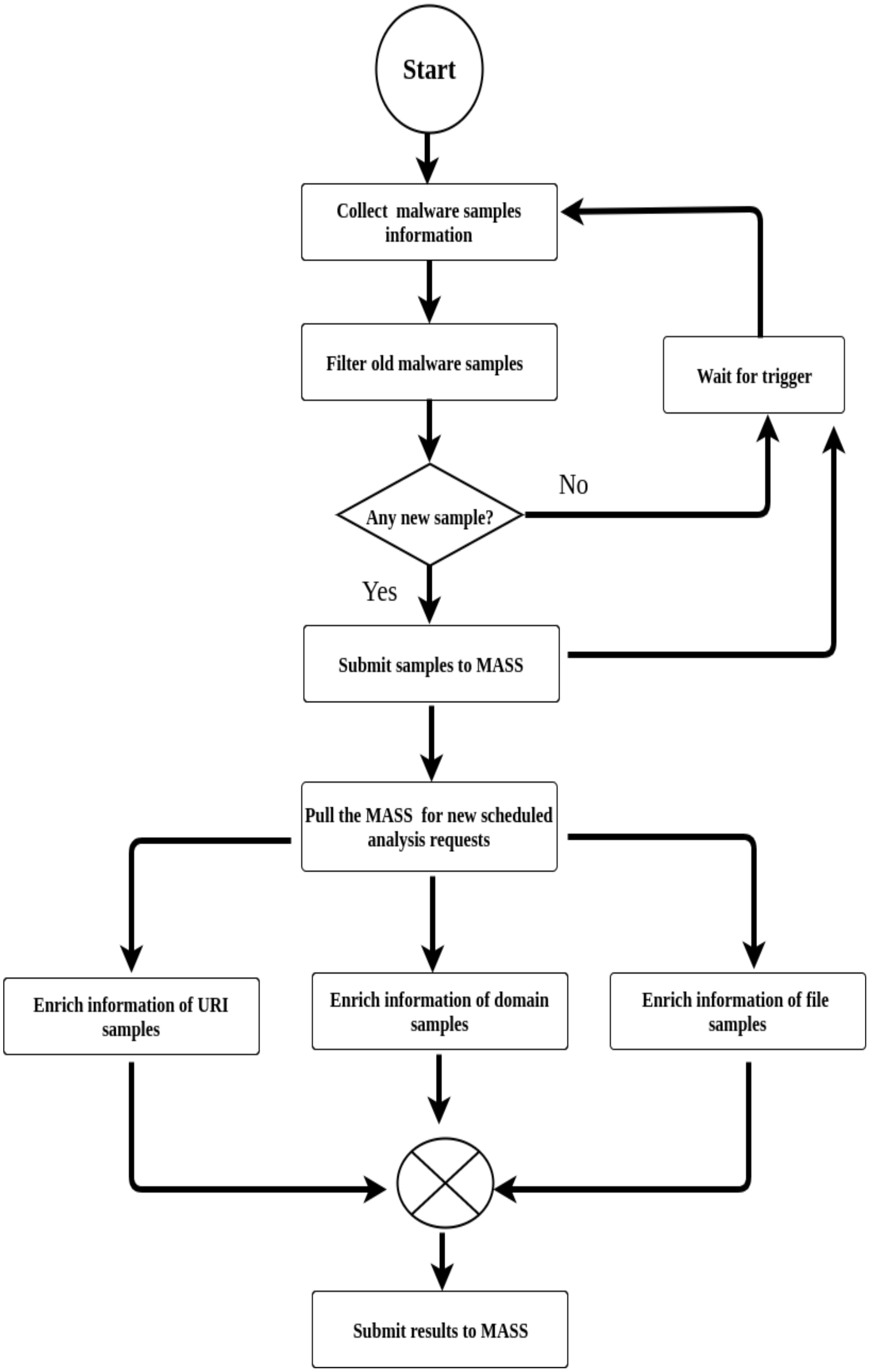
Start
Collect malware samples information
Filter old malware samples
Wait for trigger
No
Any new sample?
Yes
Submit samples to MASS
Pull the MASS for new scheduled analysis requests
Enrich information of URI samples
Enrich information of domain samples
Enrich information of file samples
Submit results to MASS

Fig. 2. Process Flow of Malware Meta Crawler.

4 Details and Implementation

- how the idea + problem is realized + code snippet - not code docs - don’t forget to include the file sample regular expression

This section describes the implementation of the various parts of the Malware Meta Crawler for MASS system. The system is implemented in **Python 3** which is available on GitLab repository[6].

4.1 Collection and Submission of Malware Samples

This section provides some of main code implementaion of the system’s part that retrieves the malware samples and sends them to the MASS server. We use the ConnectionManager of the mass_api_client to fetch content of a given URI as shown in Figure 3.

```
    tmpfile = tempfile.TemporaryFile()
    params = {’timeout’: ’60’}
    ConnectionManager().download_to_file(uri, tmpfile, False, params=params)
    tmpfile.seek(0)
    data = tmpfile.readlines()
```

Fig. 3. Fetching content of a URI Code.

The system parses the content of the different malware samples source based on the content using the following techniques:

– **Parse HTML documents**: we use the BeautifulSoup [7] python library to extract the malware data form the retrieved pages.
– **Parse XML feed**: some of the malware data is provided in XML format, the xml.dom.minidom module of Python Standard Library is used to parse and process the XML data.
– **Parse text file**: most of the malicious domains are provided in standard text file format which is parsed by our system by simply iterating over the file’s lines.

[6] https://git.cs.uni-bonn.de/lab-mass-ws1617/malware_meta_crawler
[7] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

The system pushes the fetched malware data to the MASS server using the programing APIs provided by the `mass_api_client`. Figure 4 illustrates the different procedures to submit the malware samples to the MASS server.

```

    # Create domain sample in mass
    domain = "......"
    sample = mass_api_client.resources.DomainSample.create(domain)

    # Create URI sample in MASS
    sampleURI = "....."
    sample = mass_api_client.resources.URISample.create(sampleURI)
```

Fig. 4. Submission the malware samples to the MASS server.

4.2 Analysis Clients

MASS offers the feature of scheduling analysis requests for new submitted malware samples, and the analysis systems connect to the MASS server to receive the new samples of the scheduled analysis request. To build a new analysis system the `mass_client` [8] provides the `AnalysisClient` base class, we use it in our analysis units that enrich the information of malware sample and create relation between them.

Figure 5 shows the code of the `URISamplesAnalysisInstance` class that processes all URI samples to compute their domain and create the relation between the two samples.

```
'''
This analysis system process all URI samples and find/create
 thier domain and submit the relation between them
'''


class URISamplesAnalysisInstance(AnalysisClient):

    def __init__(self, config):
        super().__init__(config)


```

[8] https://github.com/mass-project/mass_client/tree/master/mass_client

```
    def analyze(self, scheduled_analysis):

        # Recieve new scheduled analysis from the mass sever
        # and fetch the malware sample from it
        sample = scheduled_analysis.get_sample()

        domain = getDomainOfURI(sample.uri)
        # Submit or get the domain sample
        sampleDomain = mass_api_client.resources.DomainSample.\
            create(domain)

        # Create the relation between the domain and
        # the sample URI
        mass_api_client.resources.ResolvedBySampleRelation.\
            create(sampleDomain, sample)

        # submit a report
        self.submit_report(
            scheduled_analysis,
            json_report_objects={'domain_report':
                                     {'domain_url': sampleDomain.url}},
        )
```

Fig. 5. URISamplesAnalysisInstance code.

5 System Evaluation

Evaluation performance + number of samples - state what do you like to find and how? - state perfomance metric like time, memory usage, etc. - environment setup - present the results - conclude findings

6 Conclusion and Future Work

-briefly sum up what was include/done -state the overall achievement -state the future work - measure the difference time between submission time between samples.

The evaluation of our framework shows that it achieves a comparable accuracy with respect to some of the best approaches presented in the literature.

References

1. Bayer, Ulrich, et al. ”Dynamic analysis of malicious code.” Journal in Computer Virology 2.1 (2006): 67-77.
2. Kienzle, Darrell M., and Matthew C. Elder. ”Recent worms: a survey and trends.” Proceedings of the 2003 ACM workshop on Rapid malcode. ACM, 2003.
3. Symantec. Internet Security Threat Report, Vol. 21 https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf, 2016.
4. Egele, Manuel, et al. ”A survey on automated dynamic malware-analysis techniques and tools.” ACM Computing Surveys (CSUR) 44.2 (2012): 6.
5. The Malware Analysis and Storage System(MASS). URL https://github.com/mass-project/mass_server/blob/master/README.md. Accessed: 2017-04-27.
6. The MASS API Client. URL https://github.com/mass-project/mass_api_client. Accessed: 2017-04-07.
7. Manuel Egele. A survey on automated dynamic malware analysis techniques and tools. ACM Computing Surveys, to appear.
8. Li, Chao, Wei Jiang, and Xin Zou. ”Botnet: Survey and case study.” innovative computing, information and control (icicic), 2009 fourth international conference on. IEEE, 2009.
9. Spafford, Eugene H. ”The Internet Worm Incident Technical Report CSD-TR-933.” (1991).
10. Maltrieve. URL https://github.com/krmaxwell/maltrieve. Accessed: 2017-04-25.
11. Ragpicker. URL https://code.google.com/archive/p/malware-crawler/. Accessed: 2017-04-25.
12. Van Rossum, Guido. ”Python Programming Language.” USENIX Annual Technical Conference. Vol. 41. 2007.