Aalto University
School of Science
Degree Programme in ICT Innovation

Gayathri Srinivaasan

Malicious Entity Categorization using Graph modeling

Master's Thesis
Espoo, September 06, 2016

Supervisor:	Professor N. Asokan, Aalto University
Advisors:	Perttu Ranta-aho, M.Sc. (Tech.)
	Samuel Marchal, PhD, Aalto University

Aalto University
School of Science
Degree Programme in ICT Innovation

ABSTRACT OF
MASTER'S THESIS

Author:	Gayathri Srinivaasan		
Title: Malicious Entity Categorization using Graph modeling			
Date:	September 06, 2016	**Pages:**	60
Major:	Data Communication Software	**Code:**	T-110
Supervisor:	Professor N. Asokan, Aalto University		
Advisors:	Perttu Ranta-aho, M.Sc. (Tech.) Samuel Marchal, PhD, Aalto University		

Today, malware authors not only write malicious software but also employ obfuscation, polymorphism, packing and endless such evasive techniques to escape detection by Anti-Virus Products (AVP). Besides the individual behavior of malware, the relations that exist among them play an important role for improving malware detection. This work aims to enable malware analysts at F-Secure Labs to explore various such relationships between malicious URLs and file samples in addition to their individual behavior and activity. The current detection methods at F-Secure Labs analyze unknown URLs and file samples independently without taking into account the correlations that might exist between them. Such traditional classification methods perform well but are not efficient at identifying complex multi-stage malware that hide their activity. The interactions between malware may include any type of network activity, dropping, downloading, etc. For instance, an unknown downloader that connects to a malicious website which in turn drops a malicious payload, should indeed be blacklisted. Such analysis can help block the malware infection at its source and also comprehend the whole infection chain. The outcome of this proof-of-concept study is a system that detects new malware using graph modeling to infer their relationship to known malware as part of the malware classification services at F-Secure.

Keywords:	malware, classification, graph modeling, graph mining, downloader, payload, URL, file sample, graph traversal
Language:	English

Aalto-universitetet
Högskolan för teknikvetenskaper

Aalto-universitetet
Högskolan för teknikvetenskaper
Utbildningsprogrammet i ICT Innovationk

SAMMANDRAG AV DIPLOMARBETET

Utfört av:	Gayathri Srinivaasan		
Arbetets namn: Skadlig Entity Kategorisering med användning graf modellering			
Datum:	September 06, 2016	**Sidantal:**	60
Huvudämne:	Datakommunikationsprogram	**Kod:**	T-110
övervakare:	Professor N. Asokan, Aalto University		
Handledare:	Perttu Ranta-aho, M.Sc. (Tech.) Samuel Marchal, PhD, Aalto University		

Idag, skadliga program inte bara skriva skadlig programvara men också använda förvirring, polymorfism, packning och ändlösa sådana undan tekniker för att fly detektering av antivirusprodukter (AVP). Förutom individens beteende av skadlig kod, de relationer som finns mellan dem spelar en viktig roll för att förbättra detektering av skadlig kod. Detta arbete syftar till att ge skadliga analytiker på F-Secure Labs att utforska olika sådana relationer mellan skadliga URL: er och fil prover i Förutom deras individuella beteende och aktivitet. De aktuella detektionsmetoder på F-Secure Labs analysera okända webbadresser och fil prover oberoende utan med beaktande av de korrelationer som kan finnas mellan dem. Sådan traditionella klassificeringsmetoder fungerar bra men är inte effektiva på att identifiera komplexa flerstegs skadlig kod som döljer sin aktivitet. Interaktioner mellan malware kan innefatta någon typ av nätverksaktivitet, släppa, nedladdning, etc. Till exempel, en okänd loader som ansluter till en skadlig webbplats som i sin tur släpper en skadlig nyttolast, bör verkligen vara svartlistad. En sådan analys kan hjälpa till att blockera malware infektion vid källan och även förstå hela infektion kedja. Resultatet av denna proof-of-concept studien är ett system som upptäcker ny skadlig kod med hjälp av diagram modellering för att sluta deras förhållande till kända skadliga program som en del av de skadliga klassificerings tjänster på F-Secure.

Nyckelord:	malware, klassificering, graf modellering, graf gruvdrift, dataöverföring , nyttolast, URL, fil prov, graf traverse
Språk:	Engelska

Acknowledgements

I am obliged to thank a long list of people, without whom this thesis would not have been possible. First, I would like thank my advisor at F-Secure, Perttu Ranta-aho for his valuable comments and guidance for the thesis. I would also like to thank my team at F-Secure, especially Jarrod Creado for being instrumental in my day to day work for the thesis. I take this opportunity to also appreciate the help provided by Christine Bejerasco and Karmina Aquino for giving valuable comments and steering the work in the right path. I would like to thank my manager, Jukka Haapala for giving me the opportunity to have been a part of an awesome team.

Most importantly, I would like to thank my supervisors Samuel Marchal and Prof. N. Asokan, for their constant guidance and suggestions to improve my thesis. Last but not the least, my deepest gratitude to my parents and friends for their support throughout my studies.

Espoo, 06.09.2016

Gayathri Srinivaasan

Abbreviations and Acronyms

URL	Uniform Resource Locator
AV	Anti-Virus
AVC	Anti-Virus Client
AVP	Anti-Virus Product
AVS	Anti-Virus Server
APT	Advanced Persistent Threat
RaaS	Ransomware as a Service
SE	Search Engine
BP	Belief Propagation
NRS	Network Reputation Service
FRS	File Reputation Service
HTTP	Hyper Text Transfer Protocol
DNS	Domain Name Service
JSON	Java Server Object Notation
RID	Record Identifier
IP	Internet Protocol
SHA	Secure Hashing Algorithm
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
FNR	False Negative Rate
FPR	False Positive Rate
TPR	True Positive Rate
TNR	True Negative Rate
PUA	Potentially Unwanted Application
RAM	Random Access Memory
VT	VirusTotal

List of Tables

List of Figures

Contents

1 Introduction

This work is a proof of concept of using graph modeling and computation techniques for classification of unknown malware. At F-Secure Labs (hereafter referred to as Labs) which is the backend processing systems at F-Secure, millions of URLs and file samples are analyzed every single day in order to keep the F-Secure security products up to date with their reputation information and successfully protect the users from incoming threats.

A typical AVP (for instance, F-Secure SAFE) functions as follows [25]:-

- The anti-virus backend systems perform automated classification of huge number of files everyday and maintain a reputation look up service that is hosted on the cloud. This reputation lookup service is made up of the file's unique keys along with their reputation labels which may be one of the following :- trusted, malicious or unknown.
- As soon as a file is encountered on a client system, the Anti-Virus Client (AVC) locally scans the file and uses the signatures on the client to verify the file's reputation.
- If the file is not detected by the local signatures on an AVC, the file's metadata is fetched and sent to the cloud server to fetch the corresponding reputation label from the reputation lookup service. The reputation look up service on the cloud is kept update by the anti-virus backend classification systems analyzing the files and creating reputation labels for each of them.
- The client is then notified of the file verdict and if it turns out to be malicious, the user is prevented from executing it.
- If the reputation of the file is 'unknown' and can be executed, the file is sent to the backend systems for further malware analysis, which is essentially an automated and deeper investigation of the file behavior and activity in a controlled environment.
- The results of the automated analysis are updated in the reputation lookup service on the cloud server, as soon as possible, for detection of further threats.

Similary for URLs visited on the client's system, a verdict is fetched from the cloud reputation lookup service, which is kept upto date by classification processes performed on the anti-virus backend systems. Currently, the Labs' backend systems, responsible for this automated malware classification, are responsible for analyzing URLs and file samples and assigning a reputation label to them. One of the most important steps in malware analysis is executing the malware in a controlled environment called "sandbox" to capture its behavior or activity during the execution. For example, a suspicious executable file is run in a restricted environment to capture any suspicious event that it may reveal during the process. This helps the analysts to reproduce the actual malware infection and gain first-hand information on it. However, currently, these malware entities are being handled disjointedly, it is tedious for analysts to see potentially related URLs and file samples. Moreover, analyzing the URL/file content and its behavior alone is not enough to predict its maliciousness as today's malware is packed with various evasive techniques to escape prediction by anti-malware analysis and sandboxing techniques [25]. With the usage of relationships among malware, anti-malware analysts might get additional insight into its overall malware infection ecosystem to make an informed decision [25]. For instance, a malicious file is more likely to access malicious URLs than trusted ones and vice versa. If such a relationship between a URL and a file is available to the analysts, it would help the analysts, for instance, to predict if they are all involved in the same malware infection chain. In addition, it also helps to reveal anomalous behavior which might be influential in identifying potentially malicious URLs or files. For instance, an unknown executable linking to a malicious website or an unknown URL dropping malicious files are clear indicators of maliciousness.

1.1 Contributions

This work proposes a system that aims to use graph modeling and mining techniques to represent the relationships between URLs and file samples and in turn identify the maliciousness of unknown URLs and files from their relationships to known malware.

The major contributions of this work are listed below

- We forumulate the problem of identifying potentially malicious file samples and URLs by using graph modeling.
- We develop an iterative graph-based algorithm that can identify such URLs and file samples that are more likely to be malicious from their relationships to known malware.
- Using 4 months' real classification data from F-Secure, we achieve an Accuracy of 45% in classifying unknown file samples with a false positive rate of 0%. In addition, our method is also able to identify unknown URLs with an Accuracy of 80%.

1.2 Thesis Organization

The organization of this thesis is as follows. After this introduction, Chapter 2 discusses the background of malware threat landscape, current malware trends and the distribution channels. In addition, this chapter also details about the shift to malware for profit and why there is a need for malware classification methods. Chapter 3 presents the concrete problem faced by anti-malware analysts at F-Secure and the motivation behind this thesis. This chapter also discusses the list of solution requirements that the proposed system is expected to solve. Next, Chapter 4 introduces the proposed graph modeling system and discusses about graph concepts in detail. Further, Chapter 5 introduces the proposed algorithm of using graph modeling for identifying unknown malware. Then, Chapter 6 presents the results of the proposed method along with the experiments conducted with the test datasets. We also discuss certain interesting patterns in the resulting graphs obtained from the test datasets in this chapter. This chapter also evaluates the solution requirements along with the performance of the algorithm against chosen parameters and presents the results. Finally, Chapter 7 concludes the thesis with the results obtained and presents the scope for future work.

2 Background

This section discusses in detail how the threat landscape has evolved over the years, with more focus on current trends and distribution channels used. In addition, we also discuss the advancements in the field of malware classification and different methods used by the anti-virus companies in order to keep the linchpins of the web at bay.

2.1 Current Threat Landscape

Today, with more and more businesses moving to the cloud and more devices becoming connected on the web, there have been increasing concerns in the area of security and privacy. Today's web infrastructure is constantly challenged by rapidly changing threat landscape that makes it hard for the security professionals to stay up to date with the imminent threats and attacks. These cyber attacks come in various forms and are sophisticated enough to operate undetected for a long period of time. Subsequently, it demands that organizations keep their security infrastructure under check with improved threat intelligence methods to stay protected. The cyber attacks, these days, can be of any form ranging from opportunistic attacks such as phishing, spamming to more precisely targeted ones like the Advanced Persistent Threat (APT), which is an attempt to infiltrate a specific target organization for business or political reasons [5]. The goal of the attack extends beyond immediate financial gain and usually involves prolonged control of the target systems by the attackers to get as much as information on the target as possible.

2.1.1 Malware Attacks and Trends

A malware can be any type of intrusive software that brings damage to a computer system or results in theft of critical information[1]. The malware authors are highly skilled in releasing several variants of the same malware and thereby evade detection by anti-virus products[2]. They take advantage of the growing number of connected devices on the web in order to efficiently devise invincible methods of attacks. Despite the numerous forms of attacks, more often, these malicious activities on the web are linked to each other in one way or the other and interoperate to carry out their operations. It is much critical knowledge for the security community to understand how these attacks relate to each other and the common characteristics they share in order to proactively address them. Despite successfully defending umpteen threats everyday, the anti-malware analysts still need a holistic view of the malware operation infrastructures to prepare for new and incoming threats or keep the impact to minimum.

Below are some of the common trends in cyber threats for the year 2016, according to a threat report by McAfee [21]

[1]https://en.wikipedia.org/wiki/Malware

[2]http://money.cnn.com/2015/04/14/technology/security/cyber-attack-hacks-security/

- **Increase in Mobile Malware**
  Mobile devices have been potential targets for the attackers for many years, irrespective of the platform, due to their ubiquitous usage. Before the advent of the web, the primary medium of attack used to be via SMS texts. Today, due to prolific increase in third party apps on the web stores, the focus has shifted to exploiting security vulnerabilities on these applications, which causes an alarming threat to security as well as privacy of mobile users. The recent trend is to target mobile application developers and trick them into using compromised tools for app development. Moreover, the rise in the usage of mobile banking has created new attack surfaces to the advantage of the evil attackers. Almost all regular malware such as ransomware, banking trojans etc., have been created for mobile devices as well.

- **Rise of the IoT Attacks**
  The rise of the Internet of Things (IoT) not only opens up tremendous opportunities for sharing information and staying connected but also creates endless ways for the attackers to gain access to valuable information at ease. As opposed to PC's, IoT devices are not well protected end-to-end and are an easy target for the attackers [18]. In 2014, around 100,000 everyday consumer gadgets such as routers, televisions, refrigerators were hacked and used to send more than 750,000 spam emails[3]. Though IoT attacks are similar to the traditional web attacks, the impact of such IoT attacks can be hostile and tough to recover from. With more devices connected to each other, even minor security issues will cause significant damage[4]. Mobile devices are being connected to IoT devices such as fitness trackers, activity trackers etc., and hence, mobile malware also target these devices. With everyday appliances becoming connected to the web, the nature of damage that can result from these IoT attacks is not only intellectual property but also physical safety[5]. Hence, with the development of IoT devices, we should also address the challenges concerning security and continue to research and develop new approaches to ensuring safety, security, and privacy.

- **Advanced Persistent Threats**
  One of the recent threats include *Advanced Persistent Threats*, which are aimed at specific targets ranging from individuals to sophisticated government organizations with intellectual information that they cannot afford to lose. The attackers establish a backdoor to gain *persistent* access to the target system and steal key assets by managing to stay under-the-hood for a long time[6]. An instance of an APT attack was by a Russian government-backed group known as

[3]http://www.businessinsider.com/hackers-use-a-refridgerator-to-attack-businesses-2014-1?r=US&IR=T&IR=T

[4]http://internetofthingsagenda.techtarget.com/tip/Prevent-IoT-security-threats-and-attacks-before-its-too-late

[5]http://resources.infosecinstitute.com/how-hackers-violate-privacy-and-security-of-the-smart-home/

[6]https://business.f-secure.com/5-advanced-persistent-threat-trends-to-expect-in-2016/

the Dukes, which penetrated other governments and related organizations over the course of 7 years[7]. It has become an atmost necessity for the organizations and government agencies, these days, to employ sophisticated APT prevention technologies to evade intrusion or detect any suspicious activity at the earliest.

2.1.2 Malware for Profit

There has been a paradigm shift in malware authors' motivation to create and distribute malware. The first generation malware were written simply for fun or to show off programming skills and were primarily aimed at causing as much trouble to the victims as possible. These viruses would cause malfunctioning of the computers, damage hard drives and make the systems completely unusable. However, malware authors realized the opportunity to extort money by means of phishing, bank fraud, identity theft, extortion, spamming and DDoS attacks [16]. Since the focus has shifted to financial gain, the common malware these days operate under-the-hood, without announcing their presence, unlike first generation malware. They even hide their traces of activity to elude detection or disable the anti-virus software altogether[8]. Email is one of the primary vectors for spreading malware. The financial gain from malware started primarily with email spam. In most cases, compromised systems are used to send such spam emails. Although spamming has begun to drop, spammers send billions of messages every day hoping that it reaches at least small percentage of users who successfully fall prey to the hoax. Most spam emails contain malware as attachments while some invite the users to click on links which are mostly compromised and contain malicious files. Below are some of the types of malware that gain financial benefits in various ways.

2.1.2.1 Adware

Adware is a type of malware that delivers unwanted advertisements[9]. Most of them include pop-up ads on websites and software installed by the users. Adware is regarded as an alternative offered to consumers who do not wish to pay for software[10]. Mostly adware functions as a revenue generating tool for the advertisers based on the number of users clicking on the ads. While some adware is solely designed to generate revenue through ads, it is usually very common for adware to come bundled with spyware that can track user's browsing activity and steal sensitive information. Adware/Spyware bundle is the simplest form of malware that gains profit by serving users with tailored advertisements[11].

[7]http://securityaffairs.co/wordpress/40214/cyber-crime/the-dukes-apt.html
[8]https://labsblog.f-secure.com/2016/05/06/on-the-monetization-of-crypto-ransomware/
[9]https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101
[10]http://www.webopedia.com/TERM/A/adware.html
[11]https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101

2.1.2.2 Banking Trojan

There has been a prolific increase in the number of banking malware that use common tactics to steal financial information from the victims. Most of these malware leverage the security vulnerabilities in the victim's software to intercept internet banking sessions. The user is directed to a fake bank website, that imitates the original, and steal the credentials that the user inputs. The banking trojans also affect mobile phones and steal mobile banking credentials using the same techniques. The malware authors distribute these trojans via email attachments or web ads. By doing so, they reach wider audience and gain hundreds of millions of dollars from successfully deceived victims. Some examples of banking trojans include Dyre, Zeus, Dridex, etc[12].

2.1.2.3 Ransomware Trojan

Banking Trojans have been the malware of choice for cybercriminals for years. Now, the focus has shifted to a new form of money extortion called 'Ransomware'. A Ransomware is a special type of malware that encrypts and locks the files on a user's machine and demands a ransom in order for the files to be unlocked. Their medium of spreading is similar to banking trojans, where a user is duped into opening email attachments or clicking on ads. One type of ransomware activates a timer stating that a portion of your data will be destroyed every 'X' minutes if the money is not received. Another type forces the victims to purchase a program to de-encrypt the data[13]. In most cases, the attackers give the victims their files back once the ransom is paid. This is to build a sense of trust among the users that makes them pay the ransom. Ransomware is an example of malware spread using social engineering technique to deceive the victims. The latest trend in ransomware world is providing Ransomware as a Service (RAAS), which facilitates anyone on the web with minimal or no knowledge of creating malware, to deploy one[14]. The ransom can be as little as $10, paid through a premium text message or through bitcoins. Some of the significant ransomware on the web are CryptoLocker, CryptoWall, Dridex, etc[15].

2.1.3 Malware Distribution Channels

In the previous section, we discussed some of the current malware trends and attacks. In this section, we give an overview of some of the common distribution channels used by the malware.

One of the primary goals of malware authors is to distribute malware in such a way that it impacts a wider audience and gain maximum profits. Historically,

[12]http://blog.trendmicro.com/trendlabs-security-intelligence/banking-trojans-as-a-service-hits-south-america/

[13]http://blogs.mdaemon.com/index.php/ransomware-and-banking-trojans-are-big-business/

[14]http://www.businessinsider.com/ransomware-as-a-service-is-the-next-big-cyber-crime-2015-12?r=US&IR=T&IR=T

[15]http://us.norton.com/yoursecurityresource/detail.jsp?aid=rise_in_ransomware

the most common method of malware distribution was self-propagation where the malware would propagate exploiting the server-side vulnerabilities without the need of user interaction [22]. But with the advent of the web, the attention has shifted to client side vulnerabilities and social engineering for spreading infection [17]. The attackers use multiple models for distribution of the malware. The following are some of the commonly used distribution methods [22].

2.1.3.1 Drive by Downloads

This technique involves tricking the victim to execute malware by exploiting the security vulnerabilities on the software used by him. The starting point of drive by download attacks is when the user visits a website containing malicious code. In some cases, these attacks are invisible to the user and happen without the user consciously initiating the attack. The attack can be triggered by even simply viewing a webpage that hosts the malicious code. Most of the times, legitimate websites that get compromised are used by the attackers for this purpose[16]. The attackers host files called *exploit kits* on these compromised websites, that leverage the security vulnerabilities in the user's browser or any other software to start the infection. The exploit kits in turn download payloads which contain the actual malicious code. These attacks can be prevented by keeping the operating system and softwares used up to date with latest security updates[17].

2.1.3.2 Email

Most cyber attacks are spread via e-mail. Similar to drive by downloads, this method leverages the vulnerabilities in the email client to deliver malware attachments or links to malicious URLs. These in turn would install malware on the users' machines. Most sophisticated malware are spread as spam email attachments. These attachments may include executables or even document files that contain malicious macros. When an unsuspecting user opens these documents, the malicious code is enabled and the infection begins[18]. The anti-virus companies often advise users to not open emails inviting to download executables or ask for financial information. Most of these email based attacks use the technique of Social engineering to psychologically manipulate users into downloading a malicious file or visiting a compromised URL [22] or even divulging confidential information. Spamming and Phishing are the most common examples of using social engineering to steal senstive information from the victims by making them fall prey to hoaxes[19].

[16]https://blogs.mcafee.com/consumer/drive-by-download/

[17]https://www.comodo.com/resources/home/newsletters/nov-10/ask-geekbuddy.php

[18]https://www.sophos.com/en-us/security-news-trends/security-trends/the-rise-of-document-based-malware.aspx

[19]http://www.tripwire.com/state-of-security/security-awareness/5-social-engineering-attacks-to-watch-out-for/

2.1.3.3 Downloaders

This method of malware distribution uses one or more of the above techniques as the initial step of the attack. For instance, malware downloaders may be distributed to unsuspecting victims who fall prey to infected email attachments or via social engineering methods. These are mostly the first stage of the malware infection. These not-so-genuine programs on execution install multiple malware (referred to as payloads) on the fly via compromised servers. The downloader files are packed with malicious code that disables the anti-virus product's configuration or in some cases even disable the anti-virus altogether to prevent being detected [5]. They connect to compromised servers to download the actual malicious payload files, which may be any type of malware namely Ransomware, Trojan, etc. In some cases, the downloaders do not exhibit any malicious activity such as modifying the system components and are solely used as a medium to download the malicious payloads. These downloaders simply connect to multiple compromised servers which host the malicious payloads to download. Due to this unsuspecting behaviour, it is hard for the anti-virus products to detect them by analyzing just their content or behaviour. It is for this reason that attackers use downloaders as the first stage of multi-stage malware infection. Since the downloaders remain undetected, they can be used to spread additional malware payloads. This poses a challenge for the anti-malware analysts to come up with new methods to be able to detect such malware downloaders effectively and prevent any malware infection at its source.

2.1.4 Malware Downloader Lifecycle

In the bygone years, the lifecycle of a malware was relatively simple to understand. According to a recent threat report by F-Secure, there are "*four in's*" which form the basis of any infection of a target system namely Inception, Intrusion, Infection and Invasion [8].

The stages involved in a self-contained malware are briefed below and illustrated in the Figure 2.1.

1. **Inception**
   This is the entry phase of any malware to a target system where it poses any minor vulnerability that can be leveraged by an attacker. This is usually done by tricking the user into visiting harmful websites or downloading malicious payloads.

2. **Intrusion**
   During this phase, the attacker successfully takes advantage of the security vulnerabilities of the software used in the target machine and gains access to the system.

3. **Infection**
   At this stage, the malware may or may not connect to compromised servers

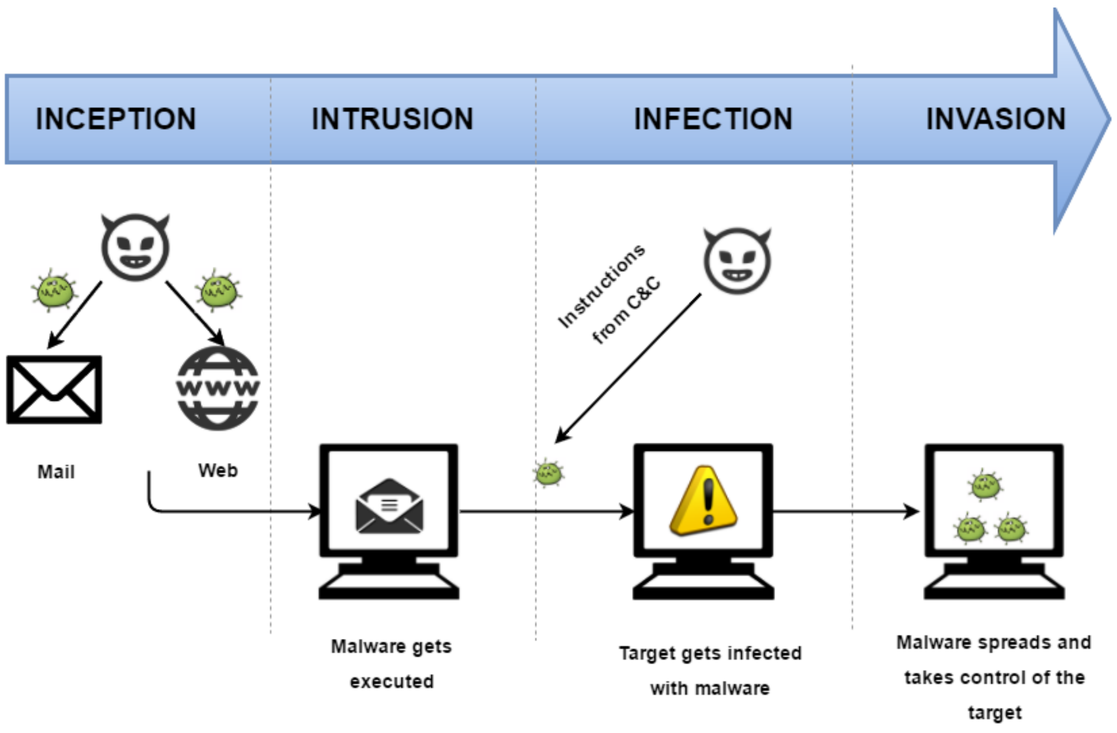


Figure 2.1: Simple Malware Lifecycle

to install malware payloads (viruses, ransomware, trojan etc) in order to compromise the target system. These malware payloads modify the system registry or configuration to result in malfunctioning.

4. **Invasion**
   Finally, the infection further escalates the effects of the malicious activity.

In contrast to the simple lifecycle of malware distribution, with the advent of new technologies on the web, the malware authors have been developing increasingly complex malware with robust business models. One such example is using multi-stage malware such as downloaders to spread the infection. The downloaders are malicious programs which form the first stage of a multi-stage infection. These programs on execution install additional malicious payloads and infect the victim's system[20]. The downloader contacts remote hosts on the fly to download the actual malware as shown in Figure 2.2. It is challenging for versatile anti-virus products to blacklist downloaders due to the fact that they are simply used as tools to spread infection and are not always malicious themselves. Due to this fact, these downloaders remain under-the-hood for a long time and are recycled to spread new variants of malware payloads. However, these downloaders are at the source of the chain of maliciousness and blacklisting them is significant to prevent any infection at the foundation.

[20]https://www.fireeye.com/blog/threat-research/2015/06/evolution_of_dridex.html

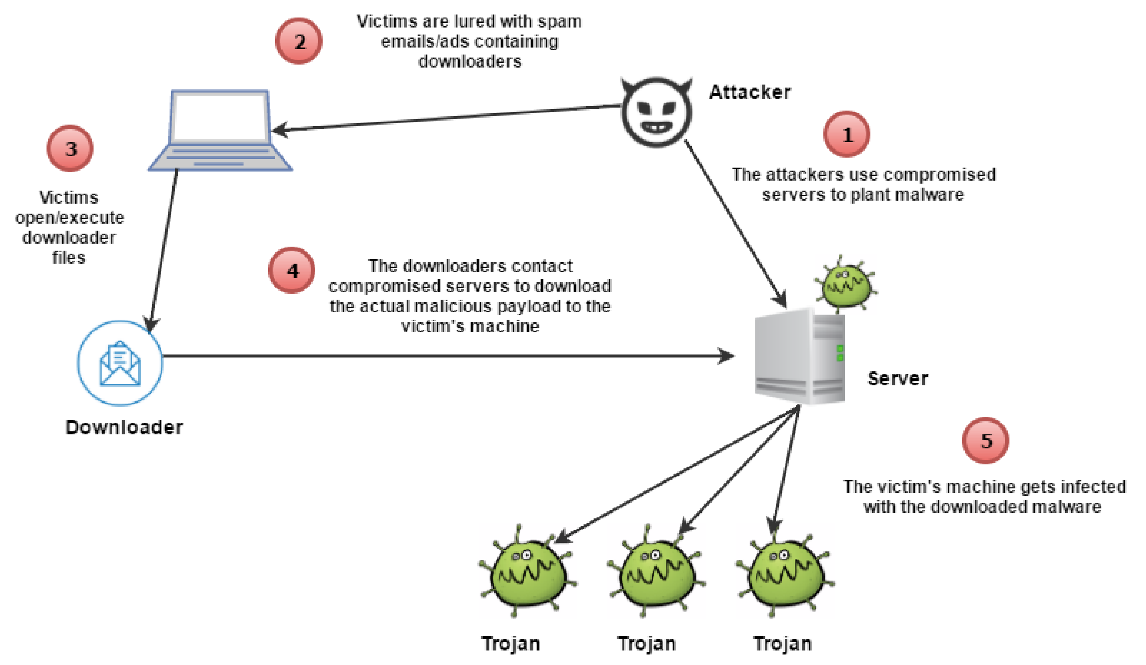

2
Victims are lured with spam emails/ads containing downloaders
Attacker
1
The attackers use compromised servers to plant malware
3
Victims open/execute downloader files
4
The downloaders contact compromised servers to download the actual malicious payload to the victim's machine
Downloader
Server
5
The victim's machine gets infected with the downloaded malware
Trojan
Trojan
Trojan

Figure 2.2: Malware Downloader Lifecycle

2.2 Malware Classification methods

There are a number of techniques used by anti-malware companies to detect new malicious URLs and file samples. These methods include techniques as simple as blacklisting to more complex methods using machine learning. Lightweight classification methods are performed on the Anti-Virus Client's (AVC) end in order to provide a reputation as soon as possible and to remain protected from existing threats. These are termed as Client-side methods which include Signature-based and Heuristic-based methods of malware classification. However, in order to prevent zero-day attacks and new malware variants, there is a necessity to apply intelligent mechanisms to predict malware [2]. This usually involves sophisticated execution and additional hardware resources that they are performed on the Anti-Virus Server's side (AVS). These methods are termed as Server-side methods which include Learning-based and Graph-based methods of malware classification.

The Figure 2.3 shows the hierarcy of the different classification methods discussed in the next sections.

2.2.1 Client-side Methods

This section gives a brief overview of some of the methods used by the AVC to blacklist malware.

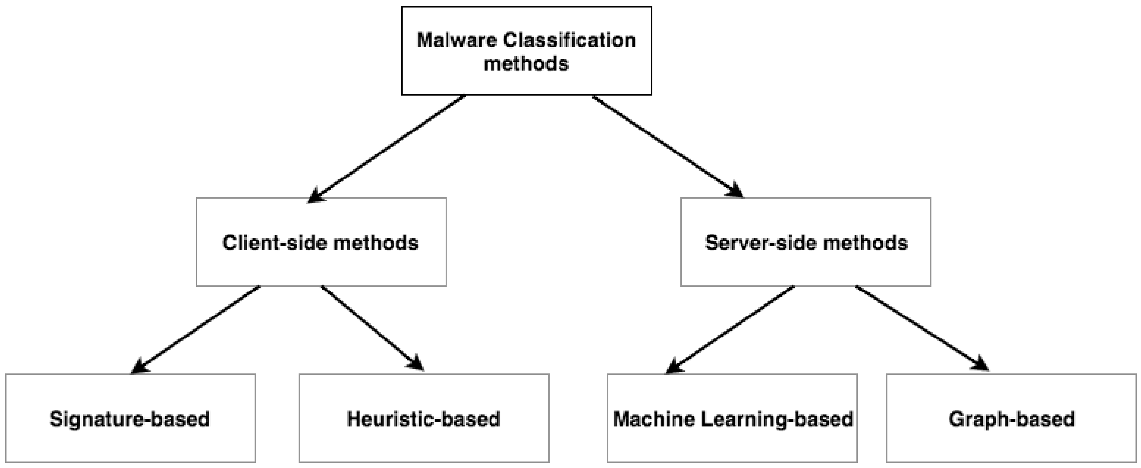


Figure 2.3: Malware Classification Methods

2.2.1.1 Signature-based detection

A malware signature is similar to a fingerprint that can be used to detect and identify a particular type of malware[21]. The AVC validates the files on the user's machine against a known list of malware signatures. This list of known malware is generally stored locally and queried by the AVC whenever needed. Today, in addition to local signatures, some of the signatures are stored on a cloud server and queried by the AVC. For instance, F-Secure has a Cloud Reputation Service called F-Secure Security Cloud, that stores information about the list of blacklisted files and URLs and keeps it upto date with new malware reputation label[7]. The attackers constantly change their modus-operandi to avoid being detected while the anti-virus companies discover new methods from time to time to prevent new threats. Once the malware gets blacklisted by the AV, the users are proactively informed and prevented from getting infected by the malware. When a user, who has the AVC installed on his machine, visits a URL or opens a file sample, its reputation is verified against the reputation service to check if it is blacklisted and thereby prevent him/her from getting infected by potential malware. Usually the blacklisting process involves not only blocking the malware outright but also labeling them into categories namely 'adult', 'violence', 'hate', etc. that will let the user make an informed decision. However, the main drawback of signature-based methods is that the anti-virus cannot detect new malware variants and zero-day threats for which signatures are not known yet[22].

2.2.1.2 Heuristic-based methods

In this method, the AVC examines the files for suspicious aspects without an exact signature match. For instance, the AVC might look for suspicious instructions or malicious code in the file which is indicative of malicious activity. The AVC might

[21] http://www.webopedia.com/
[22] https://labsblog.f-secure.com/2016/07/08/whats-the-deal-with-detection-logic/

also run the file in a virtual environment to examine the activity of the file, without noticeably slowing down the system. The biggest drawback of heuristic-based methods is that without enough information, legitimate files can inadvertently categorized as malicious[23].

2.2.2 Server-side Methods

In this section, we will discuss some of the behaviour based-classification methods used by the AVS to generate reputation labels for files and URLs. These methods require huge processing power such that they cannot be performed on the AVC. However, the classification results of these methods are constantly updated in the cloud server for the AVC to query from.

2.2.2.1 Machine Learning-based Methods

With the advancements in the field of big data, machine learning models are used to predict maliciousness of unknown files and URLs. It includes the extraction of significant features from the URLs and using intelligent classification techniques such as decision trees, Naive Bayes etc., to automatically classify the URLs into different categories [25].
Ma *et al.* [15] observed URL reputation as a binary classification problem by training the machine learning model with lexical and host-based features of URLs. A key advantage of machine learning-based methods is their ability to predict the maliciousness of new and unknown URLs without the need of analyzing the content of the URLs extensively [15].
Rieck *et al.* [20] used clustering and classification methods to group malware with similar behavior based on run-time behavior analysis. In both static and dynamic methods, machine learning models are applicable to any context in which the malware is encountered. Though these machine learning-based methods perform better than the heuristic and signature-based methods in predicting new and unseen malware, there is still a high degree of difficulty in minimizing false positives and determining which features to use in the training phase.

2.2.2.2 Graph-based Methods

The machine learning techniques are each successful in their own way in categorizing malware samples, but most of them are isolated learning techniques that consider the individual features of input samples for training and do not take into account the relationships between the samples. While analyzing individual characteristics of malware helps a great deal in malware classification, studying the relationships among them may expose the interdependence among them and improve the performance of classification [4]. With traditional methods, which use individual properties of

[23]http://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques

malware, the attackers use several obfuscation methods which can affect the feature selection process of classification based methods. In addition, these methods do not enable anti-malware analysts to get a complete view of the malware infection chain. This particularly results in labor-intensive work for the analysts to correlate the data and gather deep insight. Graph-based malware detection is based on graph theoretical approach where malware samples and URLs are modelled as vertices and the relationships between are modelled as edges. Usage of graph mining techniques can help to find malicious entities, anamolous relationships between the entities or extract similar patterns within the data. There is already abundant research on the usage of graph theory, in combination with other methods, for malware detection [4].

2.3 Existing Reputation System Overview

This section gives an overview of the existing architecture of backend malware classification systems at F-Secure. The current backend malware classification systems at F-Secure perform extensive automated analysis of file samples and URLs that cannot be handled by the anti-virus clients themselves due to the large processing power needed for the purpose. Usually, it involves executing the file samples and visiting the URLs in a simulated fashion inside sand-boxed environments[24] in order to capture their behavior. The data captured during their execution in an isolated environment provides significant information about the entities and helps in their classification. The backend classification systems at Labs are mainly composed of two separate services namely Network Reputation Service (NRS), for classification of URLs and File Reputation Service (FRS), for classification of file samples. The following sections give an overview of these services.

2.3.1 Network Reputation Service

The Network Reputation Service (NRS) is composed of a number of subsystems which are responsible for classifying URLs into appropriate categories or assigning reputation labels to them. These subsystems analyze the content of the URLs or the files downloaded from them to assign such labels. These labels include 'trusted', 'malicious' or 'unknown'. For instance, if a URL on analysis is seen to download malicious file samples, then the URL is more likely to be labelled as 'malicious'. In some cases, the URLs are also set categories namely 'hate','violence','adult', etc depending on content analysis. The NRS subsystems include sand-boxed environments which visit the URLs in an automated and controlled way to understand their behavior and record additional information for their analysis. Any executable file downloaded from the URL under scrutiny, is captured and fed back to the File Reputation Service (explained below) which is responsible for classification of the files[25].

[24]http://searchsecurity.techtarget.com/definition/sandbox
[25]https://labsblog.f-secure.com/

2.3.2 File Reputation Service

The FRS backend systems are composed of a number of automated subsystems which run through the process of classifying file samples as ’trusted’, ’malicious’ or ‘unknown’. A file sample is labelled as ’malicious’ if, on analysis, it is seen to connect to known malicious URLs. Similar to the NRS,file samples are executed in a controlled environment and their network activities are recorded. These network events might include HTTP, TCP requests etc., to several other URLs. These URLs are fed back to the NRS systems for their classification. Some file samples may download additional files (payloads) in the process which are also fed back to the FRS. Figure 2.4 represents the NRS and FRS combined system flow from end to end.

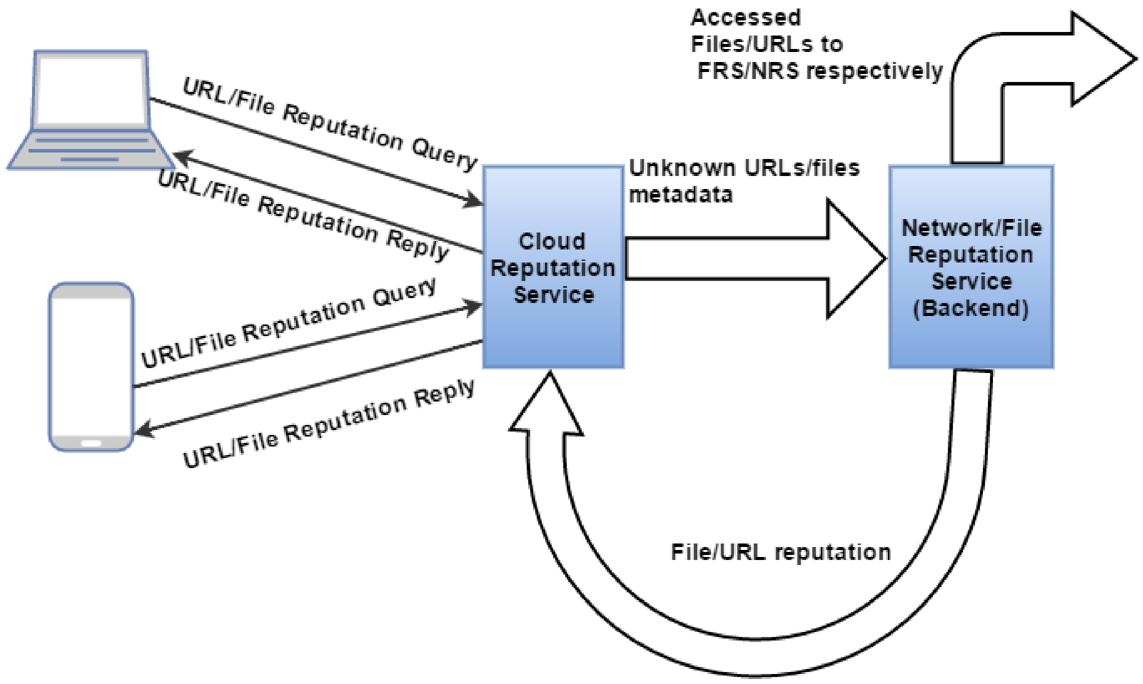

Accessed Files/URLs to FRS/NRS respectively
URL/File Reputation Query
URL/File Reputation Reply
URL/File Reputation Query
URL/File Reputation Reply
Cloud Reputation Service
Unknown URLs/files metadata
Network/File Reputation Service (Backend)
File/URL reputation

Figure 2.4: NRS and FRS system architecture

2.4 Literature Review

This section discusses some of the noticeable work in the field of malware classification. Several types of malware classification methods have been used for different purposes.

Graph-based malware classification methods have been used extensively to detect different kind of spam campaigns. Venzhega *et al.* [23] devised an effective method for Yandex Search Engine (SE) to detect malware distributors using website-file hosting relationships. The system used the log of downloads of files linked from webpages. They modelled this data as a bi-partite graph in order to detect different kinds of spam.

Our method uses a NoSQL graph database to model and visualize the malware data and relationships, which is similar to the work by Dinh *et al.* [6]. They developed a software framework for spam campaign detection, analysis and investigation of spam emails using graph database modeling. This work also involves visualization of spam campaigns and their clusters to understand the whole operating infrastructure.

One more notable research work using graph modeling and mining techniques for spam detection was by Becchetti *et al.* [3]. This work performs statistical analysis of large collection of web pages by studying graph metrics such as degree correlations, number of neighbors, etc. With the increasing usage of online services and social media, some work have focussed on tackling attacks on such services due to their wide spread usage. Huang *et al.* [9] develops a service called SocialWatch, that uses user-user social relationships to detect attacker-created accounts and hijacked accounts at a large scale. They make use of a set of graph properties including degree and PageRank for detecting attack behaviours.

There is also commendable work that are similar to our concept in categorizing malware using their relationships to other malware using graphs.

Akiyama *et al.* [1] proposed an effective blacklisting method to identify unknown URLs. The work is based on the assumption that a newly created malicious URL is in the structural neighborhood of a known malicious URL owned by the same adversary. The system uses the capabilities of a search engine in order to fetch unknown URLs in the webspace of known malicious URLs. It then uses static and dynamic content analysis on the candidate URLs to identify actual malicious URLs. This method is similar to our method in identifying unknown URLs using their relationship to known malicious URLs, although our method does not include the additional overhead of analyzing the content of the malware.

Chau *et al.* [4] presented a malware classification system that analyzes bi-partite graphs representing machine-file relationships.The system uses Belief Propagation (BP) algorithm to calculate reputation score for each of the application files that a user launches in his machine. The work uses machine learning and data mining algorithms to infer the goodness of an application. This work is highly similar to our method concept of analyzing URL-file relationships to infer the maliciousness of unknown URLs and files. Although, our method attempts to identify unknown URLs and files without the use of machine learning algorithms.

Similar to using relationships of URLs to other URLs, there is also work that leverages relationships of files to other files. Ye *et al.* [25] developed an intelligent malware classification model called Valkyrie, which utilizes both file content and relationships with one another in the form of graphs. The system is based on the assumption that two files are related if they are shared by the same client. The system uses a classification model with features from file content and relationships and is proven to have detected new malware samples not identified by other anti-viruses.

In addition to using file relations, URL relationships are also used to predict malicious communities. Li-xiong *et al.* [12] proposed a graph-based method to detect malicious URLs based on URL-URL relationships by means of the URLs visited by the same users. This method thereby detects malicious communities.

While there is good amount of work in analyzing file-file and URL-URL relationships, there is also significant work in identifying malicious downloaders and payloads using graph analysis, which is partly similar to our method. Kwon *et al.* [11] developed a graph-based model of the download activities on end-hosts to study the difference in growth patterns of benign and malicious downloader-payload graphs. This paper also applies a machine learning classification model on top of these downloader graphs to automatically learn models of malicious download graphs [11]. Our method applies similar graph modeling concepts in predicting unknown URLs and files based on the file-URL relationships. It is worth noting that simply by using graph modeling and mining techniques, our method is capable of identifying unknown files and URLs not identified by other anti-viruses.

3 Problem Statement

This chapter formulates the definite problem faced by the malware analysts at Labs and the motivation behind this work. Also, we discuss the definite requirements that the solution is expected to meet.

3.1 Problem Description

Nowadays, one of the common methods used by the malware authors to distribute malware is to install malicious downloaders via techniques such as drive-by-downloads, social engineering or regular e-mail attachments. In majority of the cases, the downloaders as such may not be inherently malicious and hence remain undetected by the anti-malware products. These software, in turn, download a variety of malicious payloads, which are responsible for spreading the actual infection to the users' machines. The downloaders also hide traces of such activity to evade detection. Due to such obfuscation behavior, these malicious downloader programs can go under-the-hood for several days or months, continuing to spread more such malware variants.

The current malware classification services at Labs use individual properties and the behavior of malware for blacklisting them but do not leverage their relationship information with one another. In addition to the classification methods used at Labs, analyzing multi-stage malware, especially downloader-payload relationships would be useful to the analysts to uncover the entire infection chain. Currently, there is no automated way of classifying malware based on the downloader-payload relationships and the analysts at Labs resort to manual work for classifying such malware.

The proposed system aims to provide an automated classification method for identifying unknown samples and URLs and thereby improve the classification performance of the current malware classification methods at F-Secure. This system functions as a server-side classification method and aims to identify unknown samples by co-relating them with known malware. This would also help the analysts understand the complete picture of the participants of an infection chain.

3.2 Solution Requirements

The goal of the thesis is to develop a system for anti-malware analysts to identify unknown samples and URLs as potentially malicious, with the help of their neighborhood information. In order to achieve the goal of this thesis, the following requirements are expected to be met.

- **Requirement R1:**

  The first requirement is to effectively report potentially malicious downloader samples and URLs using graph traversal and mining techniques with a classification accuracy of at least 50%.

- **Requirement R2:**

  The second requirement is the ability of the system to enable anti-malware analysts explore patterns and similarities between different malware families easier and faster than the current classification system.

- **Requirement R3:**

  The third requirement is the ability of the method to produce false positive rate of not more than 3%, in the process of identifying potential malware candidates. It is important to note that reporting low false negatives is not a significant requirement when compared to low false positives because high false positive rate can result in potential issues to the users and the anti-virus companies themselves. Moreover, since this solution acts as a supplement to the existing malware classification methods at F-Secure, maintaining a low false positive rate is crucial for the overall classification accuracy of the AVP.

- **Requirement R4:**

  The final requirement is the efficiency of the system in improving the classification performance of the current malware classification systems by at least 10%.

4 Data Modeling and Design

This section discusses the limitations of the existing system described in Chapter 2 and proposes our system design.

4.1 Limitations of the Existing Design

In order to evade detections by anti-virus softwares, the malware authors incorporate complex stages into the malware lifecycle making it hard for the anti-malware products to blacklist them effectively [19]. For instance, the URL blacklists can often be circumvented by distributing malware downloads from frequently changing domains. Similarly, the file blacklists can be evaded by using multiple installers or downloaders that appear legitimate but download supplementary malware payloads [10]. The existing classification systems at Labs are not effective in classifying multi-phase malware such as downloaders, since they do not examine the relationship information between the URLs and the file samples analyzed. Currently, though the NRS and the FRS systems feed related file and URL information to each other respectively, as shown in Figure 2.4, they do not leverage these relationships in order to make blacklisting decisions for malware. The following section introduces the proposed system which uses File-URL relationships in order to blacklist malware.

4.2 System Design

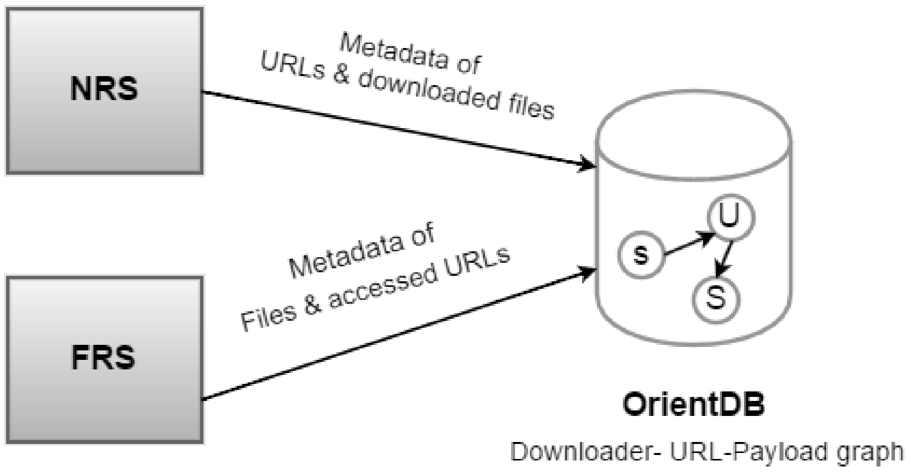


Figure 4.1: Solution Overview

The proposed solution co-relates the relationship information between the NRS and the FRS services using graph modeling as shown in Figure 4.1. The graph is modelled with files, URLs and the relationships between them which include 'File Download' activity of a URL and 'Network Connection' activity of a file sample. The

‘File Download‘ activity is a relation between a URL and the executable file that was downloaded from it. The ‘Network Activity’ relation represents the network events such as HTTP GET made by a file sample to a URL. This correlation can be represented in the form of a bipartite graph model, which can be used to determine the maliciousness of unknown malware samples from their relationship to other malware. For simplicity purposes, in the rest of the thesis, the ’File Download’ relation is called ’Download’ and ’Network Connection’ relation is called ’Activity’. The samples, URLs and their relationships are represented as a bipartite graph G (V, E) where V, a vertex, represents a URL or a file sample and E, an edge, represents the relationship between a URL and a file and vice versa. Though there are several different types of relationships between URLs and samples, in the proposed solution, they are limited to only ‘Download’ and ‘Activity’ relations.

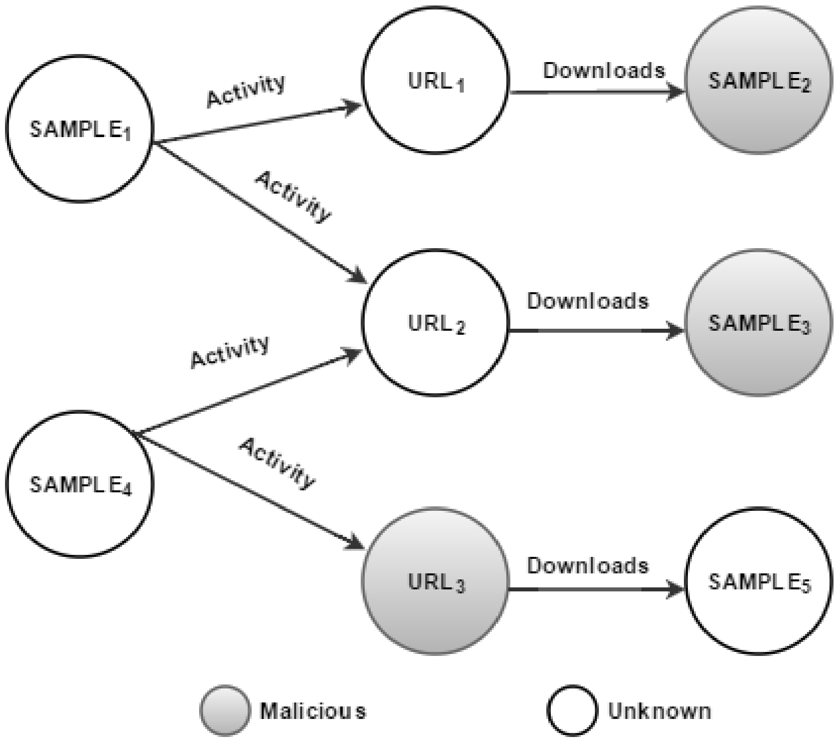


Figure 4.2: Bipartite Graph Example

The graph is bipartite as there is no link among the URLs or between the files themselves but only between the URLs and files. Figure 4.2 shows a sample bipartite graph representation of data with samples, URLs and their relationships. For instance, from the model graph, the files Sample$_1$ and Sample$_4$ are downloader samples, since they download payloads via URL$_1$ and URL$_2$, respectively. The files Sample$_2$, Sample$_3$ and Sample$_5$ are *payloads* while the URLs are called download URLs. The highlighted URLs and files are known to be malicious while the reputation of others is unknown. We can observe that the payload files, Sample$_2$ and

Sample$_3$, both malicious, are downloaded from URL$_1$ and URL$_2$ respectively, whose classification is not known.Thus, there is reason to suspect that these URLs are likely to be malicious as only malicious samples are downloaded from them. Moreover, the downloader samples Sample$_1$ and Sample$_4$ that access these malicious URLs are also more inclined to be malicious, since no clean file would access malicious URLs. Similarly, we already know that URL$_3$ is malicious. The downloader sample Sample$_4$ that connects to URL$_3$ is likely to be malicious because Sample$_4$ only talks to malicious URLs, URL$_2$ (inferred earlier) and URL$_3$. However, it is important to note that the payload Sample$_5$ may or may not be malicious since files downloaded from a malicious URL are not necessarily malware. It is also worth noting that in this work, a single download URL downloads a single payload but a single payload can be downloaded from multiple download URLs. Moreover, a payload sample can also act as a downloader and download other payloads. This is common in cases of malware that use multiple levels of distribution.

Thus, from a dataset containing downloaders, payloads and download URLs, the maliciousness of unknown downloaders and download URLs can be predicted if we know the reputation labels of the payloads. It is clear that, from the analysis of anomalous relationships between the URLs and the file samples, we can identify unknown files or URLs that are more likely to be malicious but were not identified by traditional content-based and behaviour-based methods that are currently used at F-Secure. However, manual analysis of such patterns is time consuming for the anti-malware analysts, owing to the huge amounts of data crunched by the classification systems every single day [14]. Thus, to automate this process, we propose a graph-based malware classification algorithm to classify unknown files and URLs based on their relationships with existing malware.

The proposed method is mainly based on the following two assumptions:

1. *A URL is likely to be malicious, if it downloads a malicious payload*

   A URL is likely to be malicious, if the file downloaded from it is classified as malicious. This is because no clean URL would point to a malicious payload. In most cases, legitimate servers that have security vulnerabilities are used by the attackers for spreading malware payloads. Such a URL is temporarily classified as malicious until the malicious payload is removed from it. Moreover, such URLs are used by malware downloaders as distribution mediums.

2. *A file sample is likely to be malicious, if it connects to one or more URLs that download malicious payloads* A file sample is more likely to be malicious if it connects to a malicious URL since no clean file would access a malicious URL. In most cases, the files send HTTP GET requests to webpages hosted on compromised servers to download malicious payloads into the victim's machine. These files are downloader files that may or may not be inherently malicious and simply connect to compromised servers in order to download payloads.

4.3 Graph Data modeling

Our system uses graph modeling technique where the file samples and URLs are represented as graph vertices and the relationships between them become the edges. We model this data using an open-source multimodal graph database called OrientDB, where each node and its metadata is stored in the form of a JSON document. OrientDB is an Open-Source, Multi-Model NoSQL database that combines the power of graphs and the flexibility of documents into one scalable, high-performance operational database[26].The advantage of OrientDB over other graph databases is that it is schema-less and fits our use case of modeling URLs and file samples with metadata. It also supports data visualization that can be used by the anti-malware analysts to explore malware communities and infection chains.

We describe the terminologies used to model our graph below.

1. **Sample Vertex**
   The FRS systems store the metadata of the file samples analyzed in the sandboxed environments, including the file properties and its network activity. The network events might include HTTP or HTTPS requests to other URLs which may or may not download additional malware. Subsequently, these URLs are fed back to the NRS systems for their automated analysis. It is important to note that a sample analyzed can be either a downloader or a payload.

2. **URL Vertex**
   The NRS systems store the metadata of the URLs analyzed by the sandboxed environments, including the URL properties and the files downloaded from the URL. A multi-phase malware such as a downloader may use several distribution URLs to download payloads. These payload samples are fed back to the FRS system for analysis. It is significant to note that a URL may or may not download a payload.

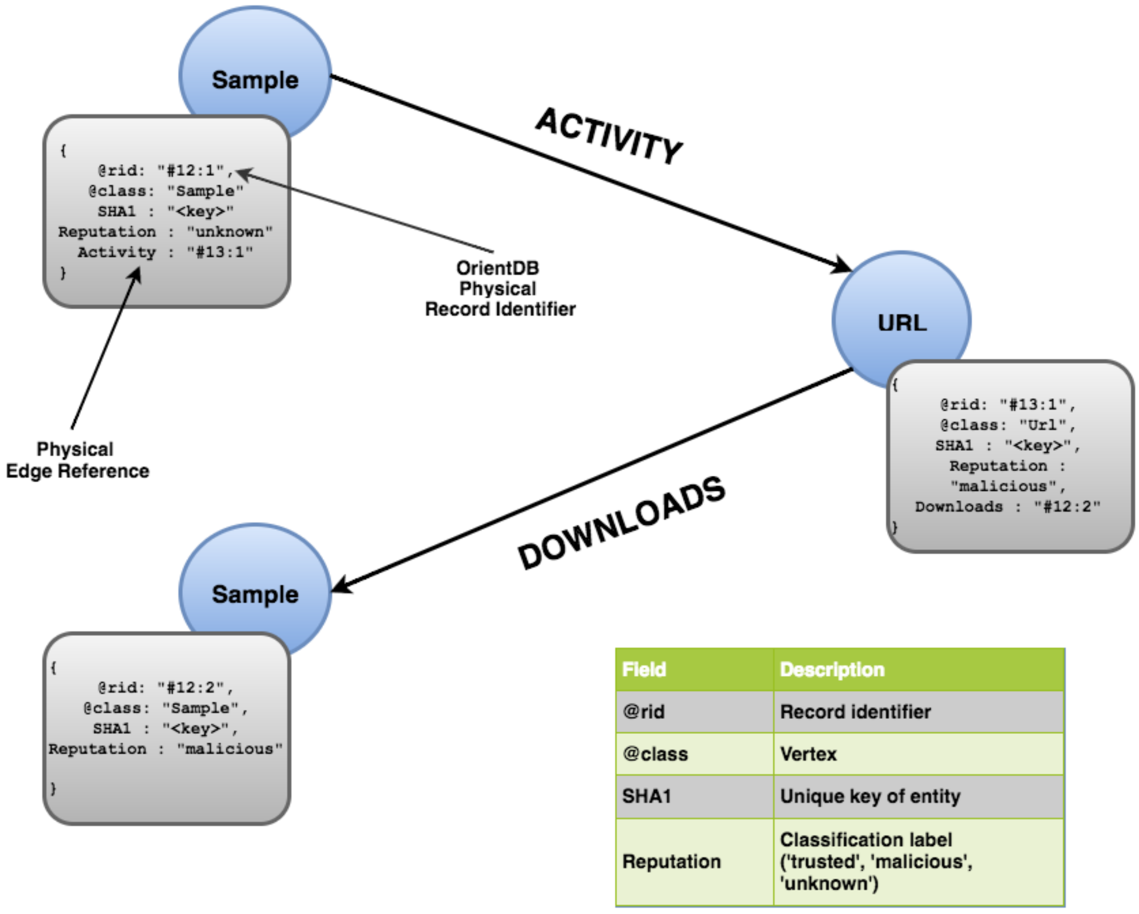
3. **Activity Edge**
   The network events such as HTTP GET or HTTP POST, created by the file samples to other URLs represent the 'Activity' edge in the graph between a downloader sample and a URL vertex.

4. **Download Edge**
   A 'Download 'edge is created in the graph between a URL and a payload sample vertex, when a payload is downloaded from the URL. However, it is not always necessary for a URL to download a payload.

[26]http://orientdb.com/

4.4 Dataset Creation

The graphs are constructed in OrientDB graph database with vertices and edges. The URLs and their metadata are fetched from the NRS systems to form the URL vertex in OrientDB. We also fetch the metadata of the payload samples downloaded from those URLs to form the 'Download' edge data model. Similarly, the downloader files and their metadata are fetched from the FRS systems to make the file vertex. We also fetch the metadata of the URLs that have been contacted by these downloaders. These form the 'Activity' edge data model in the graph.



```
{
   @rid: "#12:1",
  @class: "Sample"
   SHA1 : "<key>"
Reputation : "unknown"
  Activity : "#13:1"
}
```

```
{
   @rid: "#13:1",
   @class: "Url",
  SHA1 : "<key>",
   Reputation :
   "malicious",
 Downloads : "#12:2"
}
```

```
{
   @rid: "#12:2",
  @class: "Sample",
   SHA1 : "<key>",
Reputation : "malicious"

}
```

Field	Description
@rid	Record identifier
@class	Vertex
SHA1	Unique key of entity
Reputation	Classification label ('trusted', 'malicious', 'unknown')

Figure 4.3: Graph Data model

Figure 4.3 shows the overall data model of the graph as modelled in OrientDB graph database. The graph is made up of Sample and URL vertices, along with their metadata attributes, and the edges between them. In OrientDB, each sample and URL vertex is identified by a physical Record Identifier called RID, which is randomly generated during creation of the data. It is also important to note that the links between the vertices are stored as physical edge references to avoid the need of using JOINs as in relational databases.

5 Implementation

This section introduces the graph-based algorithm for classification of malicious downloader samples. The input data to the algorithm is a graph consisting of downloader and payload samples and URLs.

5.1 The Graph-based Method

We define the suspicion score s for a sample or a URL as a score of possibility that the sample or URL is malicious. It is a non-negative number in the range of 0-1. Based on the assumptions made in Chapter 4, we develop an iterative algorithm to calculate the suspicion scores of downloader samples and classify them as ‘malicious’, if the score exceeds certain threshold t. We also flag the download URLs as ‘malicious’ in the process. In the bi-partite graph representation, the suspicion score of a node is calculated based on the ‘malicious’ labels of its immediate neighbors. The value is calculated depending on how many of the neighbours of a node are malicious. The samples with a suspicion score of 0 are completely trusted samples while those with higher score are more likely to be malicious.

$$\text{Suspicion Score s} = \frac{\text{Number of malicious neighbours}}{\text{Total number of neighbours}}$$

We develop the graph-based algorithm 1, which takes the input graph dataset and iteratively explores the downloader samples in the neighborhood of ‘malicious’ payloads via their download URLs. We already know that each file sample and URL has a reputation label which may be ’malicious’,’trusted’ or ’unknown’. We calculate the suspicion scores of the downloader file samples based on the ‘malicious’ reputation labels of their immediate neighbour URLs. As a result, the downloader samples with a suspicion score of t or more are marked as ’malicious’. In the algorithm, the function calls to get the neighborhood URLs of a sample and neighborhood downloaders of a URL are specified in lines 9 and 13. These functions effectively traverse the immediate neighbors of a node in the graph and fetch their neighbour vertices. The pseudo code for these functions, *getNeighbourUrls* and *getNeighborDownloaders*, are not described here for simplicity purposes.

5.1.1 The Algorithm

The algorithm 1 presents the pseudocode with Graph G as the input and list of malicious downloaders and download URLs as the output. It is worth recollecting that the input graph dataset is created from the NRS and the FRS systems. The algorithm traverses the input graph starting from payload samples that are known to be malicious. It subsequently traverses the neighbor download URLs and in turn the downloader samples connected to those download URLs assigning a suspicion score to each of the downloader samples.

Thus, the algorithm outputs a list of downloader samples that are most likely to be malicious but are currently unknown to the existing malware classification systems. In addition, it also outputs a list of download URLs which are likely to be malicious. The algorithm uses known malicious samples to identify unknown URLs and file samples. Hence, given an input graph dataset, this algorithm provides a list of malicious downloaders and download URLs that are currently unknown to the existing malware classification systems at F-Secure and thereby improves the existing classification performance.

The algorithm runs until we have traversed all of the 'malicious' payloads in the input graph dataset. The result of the iterations is a set of malicious download URLs and malicious downloaders with their suspicion scores. Assuming that input graph G has *u* download URLs, *d* downloaders and *p* payloads, the running time of a single iteration of the algorithm will be *O(u + d + p)*.

Algorithm 1 To recursively fetch malicious downloader samples by traversing out of malicious payload samples in an input graph

```
 1: procedure getMaliciousDownloaders(t)
 2:     $G \leftarrow$ Input Graph
 3:     $t \leftarrow$ Suspicion Score threshold
 4:     $maliciousSamples \leftarrow$ []
 5:     $maliciousUrls \leftarrow$ []
 6:     $suspicionScore \leftarrow 0$
 7:     for each $sample \in getVertices(G)$ do
 8:         if $sample.type$ = 'payload' & $sample.reputation$ = 'malicious' then
 9:             $DOWNLOAD\_URL \leftarrow getNeighbourUrls(sample)$
10:         for each $d\_url \in DOWNLOAD\_URL$ do
11:             $d\_url.reputation \leftarrow$ 'malicious'
12:             $maliciousUrls.add(d\_url)$
13:             $DOWNLOADER \leftarrow getNeighbourDownloaders(d\_url)$
14:             for each $s \in DOWNLOADER$ do
15:                 if $s$ NOT IN $maliciousSamples$ then
16:                     $URL \leftarrow getNeighbourUrls(s)$
17:                     $s.suspicionScore \leftarrow count(URL.reputation ='malicious')/count(URL)$
18:                 else
19:                     continue
20:                 if $s.suspicionScore > t$ then
21:                     $maliciousSamples.add(s)$
        return $maliciousSamples, maliciousUrls$
```

5.1.2 Challenges in our Method

Our method is based on the assumptions that, an unknown file sample is likely to be malicious if it connects to a malicious URL which in turn downloads a malicious payload. Due to this assumption, our algorithm explores only the neighborhood of malicious payloads and ignores the payload samples for which the reputation is unknown. Also, since our method is based on relationships, there is not enough information to decide on the maliciousness of unknown URLs downloading unknown payloads. Similarly, there is little information to decide the maliciousness of unknown downloader samples that connect to unknown URLs and download unknown payloads. Hence, this method works the best, if the proportion of malicious payloads in the input dataset is significant enough to make a decision. Moreover, the suspicion score threshold of t, used for experimental purposes, is not a fixed number and can be varied to decide which score brings the best results.

One more challenge of the method is that since it works on static input dataset, there is a risk of using outdated malware reputation data, especially that of URLs. We know that a malware's reputation can change from time to time. For instance, a URL which is currently malicious can become trusted once the infection is removed and vice versa. Hence, there is a need for the algorithm to fetch real-time information than use static data to make the decisions.

6 Results and Evaluation

This section explains in detail the experiments conducted against the proposed algorithm with real malware dataset from F-Secure.
The test dataset used for the experiments consists of real malware data from F-Secure. F-Secure malware classification systems analyze and classify millions of file samples and URLs everyday in order to protect their users from imminent threats. This malware data is collected in the form of file samples, URLs, emails, IP addresses, to name a few. F-Secure maintains a blacklist of the malicious URLs and file samples and keep them up to date with several malware detection and analysis methods. If the users of F-Secure security products come across the blacklisted malware samples or URLs, they are successfully protected from falling prey to the infections. These samples are gathered from honeypots, customer submissions and from partner sample feeds such as those provided by VirusTotal. VirusTotal[27] is an online service by Google that has the reputation information of malicious files and URLs aggregated from the scan results of over 55 anti-virus engines, website scanners and security companies. Given a unique SHA1 value (of a URL or a file sample), VirusTotal will provide results from multiple anti-virus vendors whether the particular file or URL is malicious or not.

In this section, we conduct two sets of experimental studies using our malware datasets obtained from F-Secure backend systems to fully evaluate the efficiency of the proposed algorithm:

1. In the first part of experiments, we evaluate the efficiency of the proposed graph based malware classification method.

2. In the second part of experiments, we present some common patterns that are visible in the graph model of malware data.

6.1 Evaluation metrics

We measure the malware classification accuracy of the proposed system using the following evaluation metrics:

- **True Positive Rate (TPR):** The ratio of malware samples (both files and URLs) correctly classified as 'malicious' by the system. It is also known as *Recall*.
$$TPR = \frac{TP}{TP + FN}$$

- **True Negative Rate (TNR):** The ratio of samples (both files and URLs) correctly classified as 'not malicious' by the system.
$$TNR = \frac{TN}{TN + FP}$$

[27]https://www.virustotal.com/en/documentation/

- **False Positive Rate (FPR):** The ratio of samples (both files and URLs) misclassified as 'malicious' by the system.

$$FPR = \frac{FP}{FP + TN}$$

- **False Negative Rate (FNR):** The ratio of malware samples (both files and URLs) misclassified as 'not malicious' by the system.

$$FNR = \frac{FN}{FN + TP}$$

- **Accuracy:**

  Accuracy is the proportion of true results, either true positive or true negative, in a population. It measures the degree of veracity of a diagnostic test on a condition[28].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision:**

  Precision defines how many of the returned samples are correctly classified as malicious out of the total number of files or URLs (both legitimate and malicious) identified as malware. It is otherwise known as positive predictive value[29].

$$Precision = \frac{TP}{TP + FP}$$

6.2 Dataset Description and Ground Truth Labelling

We used the malware data analyzed by F-Secure between March and June, 2016 for this experiment. The reason for choosing the timeline is to consider only the latest malware data for the experiments, in order to focus on the new and prevalent threats and ignore information which may be outdated. All these data have been analyzed by sandboxed environments at least once during the specified timeline and a reputation has been assigned to each of them if found to be malicious, clean or unknown. The dataset obtained from F-Secure backend systems consists of 33,368 unique file samples and 207,865 unique URLs. Out of these numbers, we only consider the file samples that are either a 'downloader' or a 'payload' and the URLs that are download URLs. This amounts to 1,735 unique downloader file samples and 199 payload file samples. The number of 'download URLs' which connect these downloaders and payloads is 452. Although, the test dataset does not

[28] http://www.lexjansen.com/nesug/nesug10/hl/hl07.pdf
[29] https://en.wikipedia.org/wiki/Precision_and_recall

cover all malware families extensively, it represents some of the significant threats that were prevalent during this time period. These statistics are tabulated in Table 6.1

It is also worth noting that the dataset comprises 1,551 malicious, 25 clean, 357 unknown file samples (both downloaders and payloads) along with 407 malicious, 38 clean and 7 unknown download URLs. All the experimental studies have been conducted in a cloud environment with Debian GNU/Linux 8.2 operating system with 2 GB of RAM and 20GB of disk space.

Type	**Numbers**
Unique files	33,368
Unique URLs	207,865
Unique downloader samples	1,735
Unique payload samples	199
Unique download URLs	452

Table 6.1: Statistics of the test dataset

To check the ground truth of the downloader file samples classified by the algorithm, the results of the experiments are verified against F-Secure malware classification system and VirusTotal to confirm if they are in fact malicious. First, the SHA1 values of the downloader samples identified as malicious by our algorithm are looked up against F-Secure malware classification system. If not identified by F-Secure, we use VirusTotal to verify the maliciousness of a particular file sample. If at least 5 anti-virus engines classify the sample as malicious, the sample is indeed considered malicious. This is to make sure that false positives by other anti-virus engines are not considered as ground truth [13].

6.3 Accuracy of the Method

This section discusses the experiments conducted with the input dataset and calculates the accuracy of the method in each of them.

6.3.1 Classification of downloaders

In F-Secure malware classification system, there are many newly found malicious URLs every day via traditional methods. The most dangerous URLs that need attention are the ones, which when visited, download payloads that host malicious code and affect the victim's system. Some examples of malicious URLs found by traditional malware classification methods are shown in Table 6.2.
In the first set of experiments, we use the algorithm (introduced in previous section) to identify 'potentially suspicious' downloader file samples that download malicious payloads through one or more download URLs.

For performing this experiment, the reputation labels of the downloader file samples were removed and passed as input to the algorithm along with their relationships with other URLs and files. The algorithm identifies the downloader samples that are 'potentially suspicious' along with their suspicion scores based on their neighborhood. These malware downloaders are the source of any infection and most of them do not exhibit any malicious behavior themselves and hence remain undetected for a long time. By exploring the neighborhood of the downloader samples, we assign a suspicion score to each of the downloader samples based on the reputation labels of their neighbours. The unlabeled downloader samples with a suspicion score above a certain threshold t are filtered out as 'potentially malicious' from the input dataset. For the first iteration of the experiment, we set the suspicion score threshold as 0.3 and study the results.

Malicious URLs
http://findjj12.3eeweb.com/popup.exe
http://cos.myqcloud.com/11001025/yun/AtShz.rar
http://mpower.lv/system/cache/word.exe
http://secuservernet18.com/dds/t58.exe
http://down.fyeshs.com/jd_rg0cfc_xax7009_1.exe

Table 6.2: Sample malicious URLs

From the input dataset, out of the 1,735 unlabeled downloader samples present in the dataset, 1,052 samples were marked as malicious by the algorithm with a suspicion score of 0.3 or more while 683 samples were classified as trusted. The reputation of these samples were verified against the ground truth and substantiated that they were actually malicious. With these results, we check the ground truth to verify the maliciousness of all the samples. About 84 of the 1,052 samples were not identified by F-Secure engine but from VirusTotal and manual analysis. Hence, for a suspicion score threshold of 0.3, this method increases the existing backend malware classification rate of F-Secure by about 8%.

It is worth noting that there were 10 downloader samples listed in Table 6.3 with a suspicion score of more than 0.3 but were identified neither by F-Secure engine nor by any anti-virus on VirusTotal. These were identified as 'Potentially Unwanted Applications' (PUA) by manually analyzing them with the help of anti-malware analysts at Labs. These samples are perfect examples of malware samples that do not exhibit any malicious behavior as such but use compromised servers as their distribution medium to spread malware. This proves that the algorithm was able to find potentially malicious files not identified by any other anti-virus engine.

We repeat the experiment with a higher suspicion score threshold of 0.4 and analyze the results. For the new suspicion score of 0.4, out of the 1,735 unlabeled downloader samples found in the dataset, 746 samples were marked as malicious by the algorithm while 989 samples were classified as trusted. The results of this

Downloader SHA1	**Suspicion score (0-1)**
6b2e3d5fd183d96ea2027b8eb75e28ce9519669f	1
1fc2c3f2d53f69c9c946f7491d06db2f5f177aad	0.6
f1b501c52c4f523fc3df00b54cfd4af21531b0f8	0.4
1f96645c825a6a252e6de29f0cb3a140306be4a5	0.6
918389409efca718e723aa0dcf5219214935f820	0.6
0e1f282734f3b868081f04e743fcc7ef83b1d718	0.6
afcce5f141e6406d70d0c2754646bc0871c3c4fa	0.6
08edc726ad08a6e3e87eaa864e03f7305290315e	0.6
2f4e4757e18ac7e1b9a4262607fc504bdba4593c	0.6
4132e710e3dc3a902d79379d1c563f89c2c9c590	0.6

Table 6.3: Unknown malware downloaders

experiment are tabulated in Table 6.4 for both the suspicion scores. The maliciousness of these samples were verified against the ground truth and the metrics for both iterations are listed in Table 6.5.

Type	**Suspicion score (0.3)**	**Suspicion score (0.4)**
Total unlabelled downloaders	1,735	1,735
Malicious	1052	746
Trusted	683	989

Table 6.4: Results of the experiment

Downloaders classified as 'malicious'	**Suspicion score (0.3)**	**Suspicion score (0.4)**
F-Secure	967	665
VirusTotal (VT)	74	71
Manual analysis	10	10
By our method = Sum(F-Secure + VT + Manual analysis)	1051	746
False Positives (FP)	1	0

Table 6.5: Statistics of the classification

It it also worth highlighting the fact that using a suspicion score of 0.3 returned a low FPR of 2.3% and an accuracy of 0.62. For a suspicion score threshold of 0.4, this method increases the existing malware classification rate of F-Secure backend classification by about 10%. The overall accuracy metrics for both iterations of this experiment are listed in Table 6.6. From the table, it is clear that the FPR has dropped to 0 and the Precision is 100%

Measure	Suspicion score (0.3)	Suspicion score (0.4)
True Positive Rate	62.1%	44.1%
True Negative Rate	97.7%	100%
False Positive Rate	2.3%	0%
False Negative Rate	37.8%	55.8%
Accuracy	62%	45%
Precision	99.9%	100%

Table 6.6: Accuracy metrics

We also note that with an increase in threshold of the suspicion score, though the false positives have become null, the false negatives have risen to 55%. For anti-virus companies, it is important to measure not only their detection capabilities but also the trustworthiness. One of the important aspects is the anti-virus product's tendency to flag clean files as infected, which causes greater concerns and results in bad reputation to the company. These false alarms can result in more damage than a real infection. Hence, there is a small tradeoff to stay as low as possible on the false positives ratio even if that causes an increase in false negatives. Moreover, this method, when used in addition to the already existing classification methods to classify malware samples, results in significant increase in overall classification accuracy of the anti-virus engine. This method acts supplementary to the existing server-side classification methods at F-Secure that process millions of URLs and files everyday and assign a reputation to each of them.

6.3.2 Classification of the download URLs

In addition to malicious downloader samples, our algorithm can also find possibly infected download URLs which are used as a medium to spread the malicious payloads. These download URLs are mostly short lived and are used just for the purpose of spreading malware payloads. Usually, once anti-virus engines start to blacklist these URLs, the attackers take them down and create more new URLs and every day, more and more new malicious URLs are being created[30].

In this experiment, we identify unknown download URLs as malicious or trusted and verify the results against the ground truth. Table 6.7 lists the results of the experiment of identifying unknown download URLs.

We verified these results against the same ground truth data which includes VirusTotal and F-Secure malware classification system. From Table 6.8, it is clear that about 331 of the 337 URLs were already identified by F-Secure engine and 1

[30]http://arstechnica.com/security/2012/06/google-detects-9500-new-malicious-websites-daily/

Type	Numbers
Total unlabelled download URLs	452
Malicious	337
Trusted	115

Table 6.7: Results of the experiment- download URLs

URL from identified from VirusTotal. This method increases the existing backend malware classification rate of URLs at F-Secure by about 0.2%.

Download URLs classified as 'malicious'	Numbers
F-Secure	331
VirusTotal (VT)	1
Manual analysis	0
By our method = Sum(F-Secure + VT + Manual analysis)	332
False Positives (FP)	5

Table 6.8: Statistics of the URL classification

The overall accuracy metrics of the classification of URLs are mentioned in Table 6.9. This experiment returned a high false negative rate which implies that the algorithm classifies more malicious URLs as trusted. This is because the algorithm explores the neighborhood of only malicious payload samples to classify malicious URLs associated with them, whereas the input test dataset from F-Secure at that point of time has a number of 'unknown' payloads resulting in false negatives. This can be reduced if the reputation labels of the unknown payloads is known before the experiments.

True Positive Rate	81.3%
True Negative Rate	90.6%
False Positive Rate	10.4%
False Negative Rate	18.6%
Accuracy	80%
Precision	98.5%

Table 6.9: Accuracy of the URLs predicted

Overall, it is clear from the above experimental results, that the proposed method works well to identify unknown downloaders, if the reputation of the payload samples is known. For unknown payload samples, there is little information to make a decision on the download URLs or the downloader samples. With increase in classification of malicious payloads or with the addition of metrics such as WHOIS and domain hits information, we can further improve the True Positive Rate of classification of both the downloaders and the download URLs. Hence, this method could result in increase in the malware classification accuracy, when used in combination with other traditional methods.

6.4 Exploration of Malware Patterns

Malware attackers use several obfuscation methods such as domain generation, domain fluxing, etc. to make detection of their infrastructure even harder for the anti-virus products [24]. Some malware leverage the security vulnerabilities found on existing innocent servers to distribute the actual malicious code. This usually makes it hard for the anti-virus products to detect malware when they are hosted on legitimate servers. Tables 6.10 and 6.11 list a snapshot of the download URLs from the test dataset. These URLs are used by the malware families called 'Locky' and 'Dridex' to distribute malicious payloads. Locky is a ransomware that encrypts the victim's machines and demands a ransom in order to retrieve the victim's files. These websites have no similarities between them except the fact that they all have been compromised and leveraged by the attackers for spreading the malware infection. These set of download URLs are a perfect example where the same malicious payload is being hosted on multiple websites and are downloaded via multiple downloader samples.

OrientDB has a visualization interface[31] that can be used to explore data visually with the use of simple SQL-like queries or manual traversal of the data. Figure 6.1 is a sample snapshot of some Locky downloaders connecting to 3 compromised download URLs, all of which download the same malicious payload. This shows how Locky ransomware operates via compromised servers to spread infection. The downloaders used by Locky are Microsoft Office Word documents, which when opened, use macro scripts to contact these compromised servers. When a user requests a webpage found on such infected servers, they are at high risk of downloading file samples with malicious code and getting infected[32]. It is worth mentioning that for space constraints, Figure 6.1 represents only a small portion of the total number of Locky malware samples that are present in the test dataset.

On the other hand, Dridex is a banking trojan which also follows the same technique but steals banking information from the victims. Dridex operates similar to Locky in using compromised servers to spread malware[33] and this is obvious from the graph exploration snapshot of Dridex shown in Figure 6.2.

Hence, in addition to identifying individual malicious downloaders and download URLs, our method is effective in identifying the infrastructure of malware distribution by means of graph exploration. For instance, there are similar patterns of distribution between Locky Ransomware and Dridex banking trojan. Both these malware campaigns happened at different time periods[34] but their behavior and distribution

[31]http://orientdb.com/docs/2.1/Home-page.html

[32]http://www.symantec.com/connect/blogs/locky-ransomware-aggressive-hunt-victims

[33]http://researchcenter.paloaltonetworks.com/2016/02/locky-new-ransomware-mimics-dridex-style-distribution/

[34]http://www.pcworld.com/article/3033886/locky-ransomware-which-infects-like-dridex-hits-

method is similar which is evident from Figures 6.1 and 6.2. With the help of the visualization interface of OrientDB, exploration of patterns in large amounts of data is easier and faster when compared to manual co-relation of information from different reputation services. With current system, the above scenario of exploring similarities between two different families of malware would require intense manual effort and analysis by the anti-malware analysts when compared to using graph modeling and visualization.

Distribution URLs of Locky
http://www.rbb.ru/grh5444tg
http://ernetfree.net/grh5444tg
http://denzil.com.au/grh5444tg
http://habr.net/grh5444tg
http://www.haldensleben-web.de/grh5444tg

Table 6.10: Distribution URLs for Locky

Distribution URLs of Dridex
http://www.apparelbycheryl.com/9uh87g756
http://ernetfree.net/grh5444tg
http://www.studiopanella.it/9uh87g756
http://www.kosmetikafm.wz.cz/9uh87g756
http://www.pececitos.com/9uh87g756
http://www.cdc-ccd.org/9uh87g756

Table 6.11: Distribution URLs for Dridex

the-unlucky.html

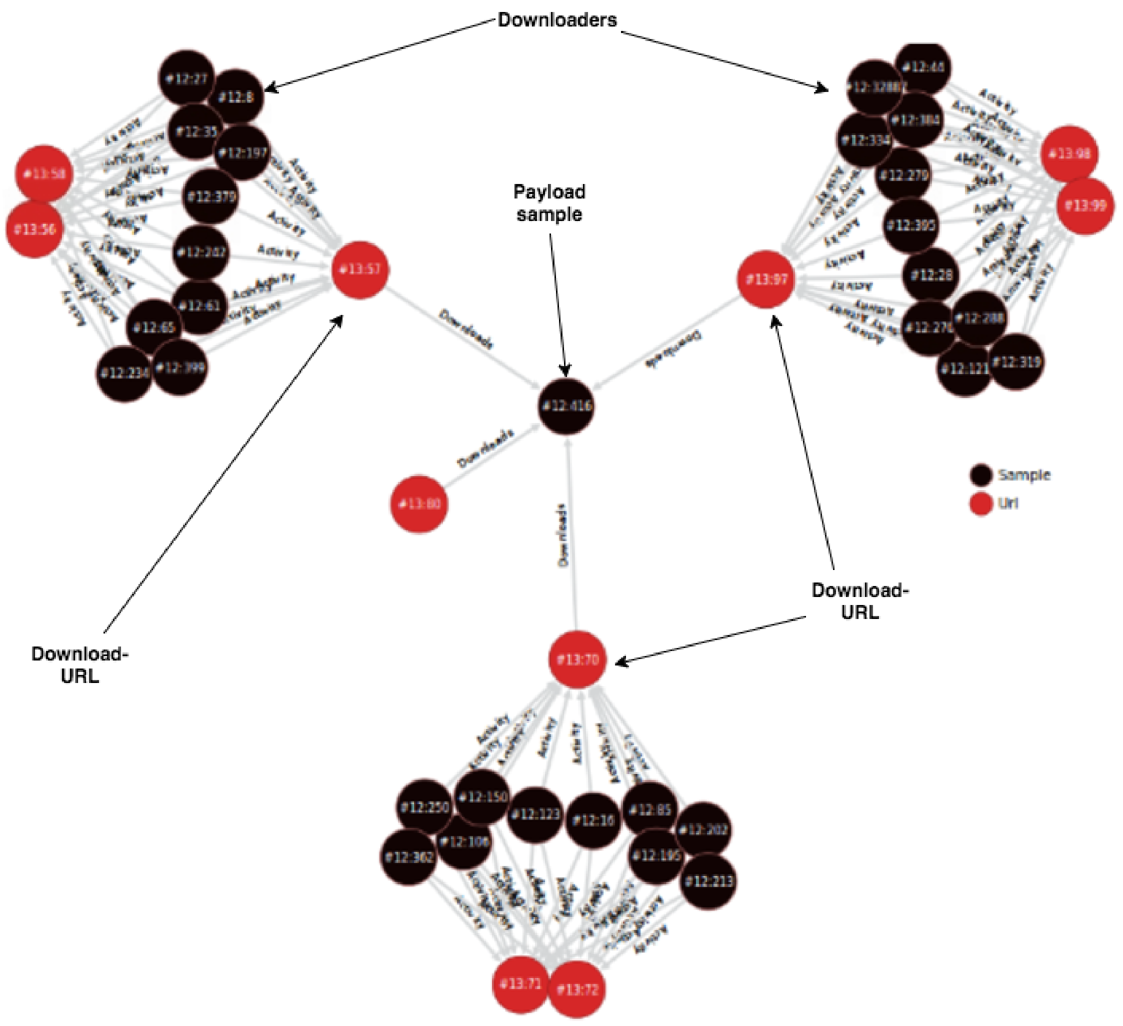


Figure 6.1: Locky Ransomware distribution

6.5 Evaluation of Solution Requirements

As mentioned in Chapter 3, our system is expected to meet the following 4 requirements. Here, we evaluate the four requirements with the results achieved.

- **R1**

  From our experimental results, it is evident that the algorithm was able to identify unknown downloaders with an accuracy of 62% and unknown URLs with an accuracy of 80%. Hence, the requirement R1 is met.

- **R2**

  From the graph patterns displayed in Figures 6.1 and 6.2, it is evident that the system enables anti-malware analysts explore large amount of data via the visualization interface faster than the current method. This helps anti-malware analysts at Labs uncover patterns in large amounts of data with less effort.

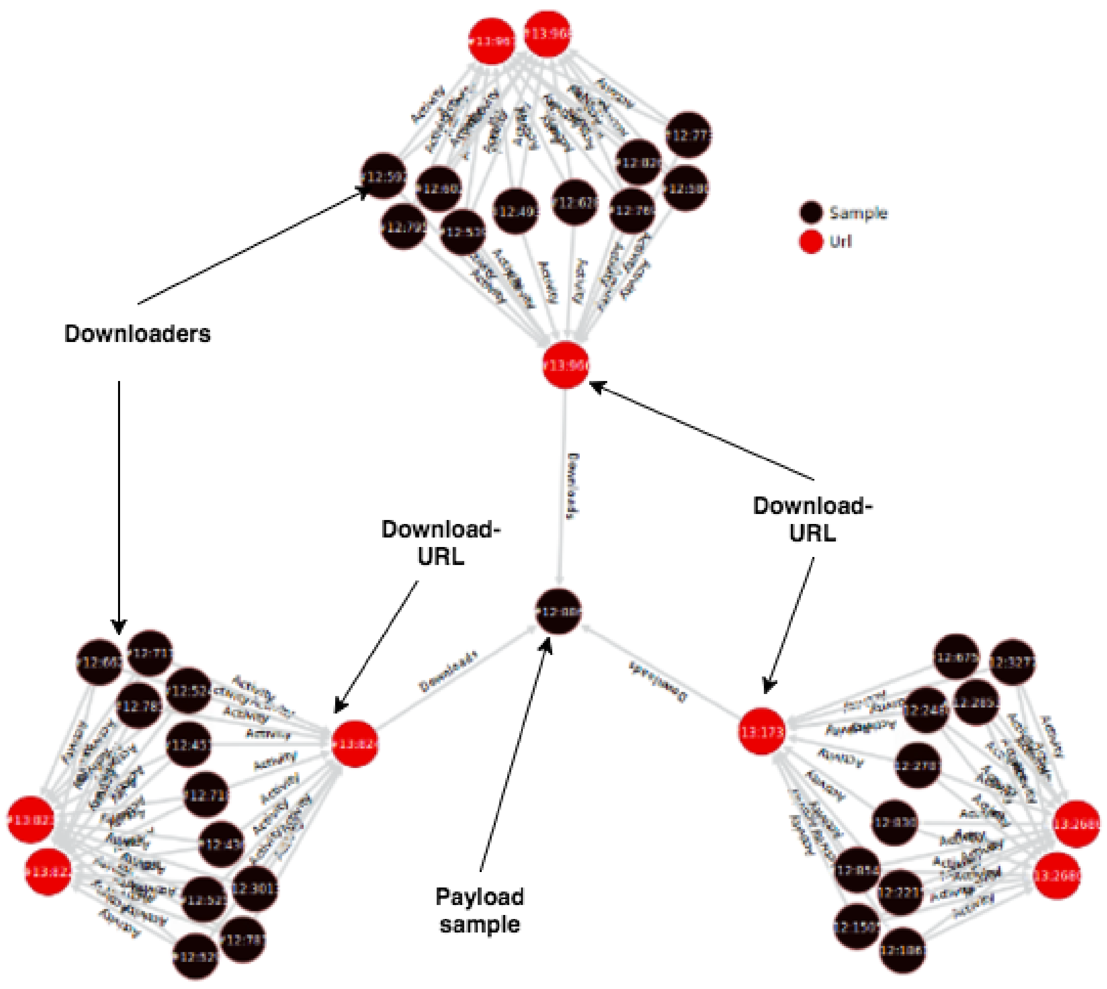


Figure 6.2: Dridex malware distribution

- **R3**

  In the experimental results for identifying unknown downloader samples, we received an FPR of 0% and a high FNR of 55.8% , when the suspicion score was set to 0.4. This proves that the method is able to fulfil requirement R3, which expects the solution to have an FPR of not more than 3% along with significant classification accuracy.

- **R4**

  Finally, the system is able to identify some malware downloader samples that are not identified by the existing classification methods at F-Secure, thereby improving the current classification performance by 10% as stated in requirement R4.

Thus, the system is able to fulfil all of the four solution requirements defined in Chapter 3.

6.6 Performance Evaluation

In this section, we evaluate the performance of the algorithm with different input metrics. We study the running time of the algorithm for various in-degree values of the payload vertices and then, we repeat the same evaluation for the in-degree of the download URL vertices.

6.6.1 Payload In-degree Analysis

For the first evaluation experiment, we vary the in-degree of the payload nodes in the graph to study the performance of the algorithm. We pick the payload's in-degree measure because our method traverses the download URLs of each payload and in turn, the downloader samples of each download URL. Hence, the algorithm is majorly dependent on the number of in edges to the payload samples. As already mentioned in Section 5, the running time of our algorithm is O (u + d + p), where u is the number of download URLs, d is the number of downloaders and p is the number of payloads. This implies that the running time depends on the in degree of the payload samples as the higher the in-degree, the higher the number of URL nodes to traverse in the graph. We evaluate this claim by running the algorithm repeatedly for different payload samples in the increasing order of their in-degree and document the results.

Degree Range	**Number of Downloaders**	**Run time (seconds)**
1-3	185	1.417
4-7	305	1.751
8-15	388	1.779
16-30	674	6.462
31-60	264	1.674

Table 6.12: Payload in-degree Vs Run time

From Table 6.12, we note that with increasing in-degree, the run time of the algorithm increases linearly. We notice that the number of downloaders increases exponentially when the in degree is increased from the range 8 -15 to 16-30. This subsequently results in the exponential increase in the run time of the algorithm. We also note that there are less number of downloaders for the payloads with in-degree between 31 and 60. Due to this, the running time of the algorithm for these nodes falls to a lower value. Figure 6.3 shows the behaviour of the algorithm when we vary the in degree of payload samples from 1 to 60.

6.6.2 URL In-degree Analysis

For the second evaluation experiment, we vary the in-degree of the download URL nodes in the graph to study the performance of the algorithm. In addition to payload

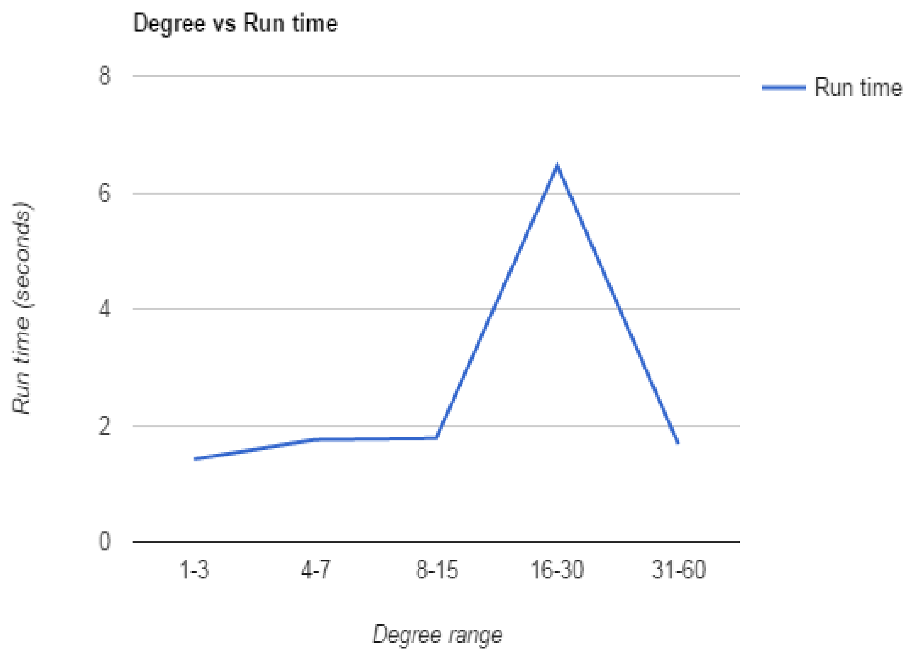


Figure 6.3: Payload In-degree vs Run time

in-degree, URL in-degree also decides the number of nodes analyzed in the graph. This implies that the algorithm running time increases as in-degree of the download URLs increase. We evaluate this claim by running the algorithm repeatedly for different download URLs in the increasing order of their in degree and document the results.

Degree Range	**Run time (seconds)**
1-3	1.906
4-7	2.131
8-15	2.159
16-30	2.469
31-60	3.581

Table 6.13: URL In-degree Vs Run time

From Table 6.13, we note that with increasing in-degree of download URLs, the run time of the algorithm increases linearly. Figure 6.4 shows the behaviour of the algorithm when we vary the in-degree of download URLs from 1 to 60. This is due to the fact that when a download URL vertex has more incoming edges, the algorithm would have to traverse more edges.

In this section, we evaluated the algorithm with different input measures to study how the algorithm behaves. The results show that the performance of the algorithm is highly dependent on the number of payloads and downloaders that are present in the input dataset. With smaller in-degree values for the payloads, the algorithm takes very less time in traversing the nodes and vice versa. Similar behaviour is observed for a smaller out-degree of the download URL nodes as well.

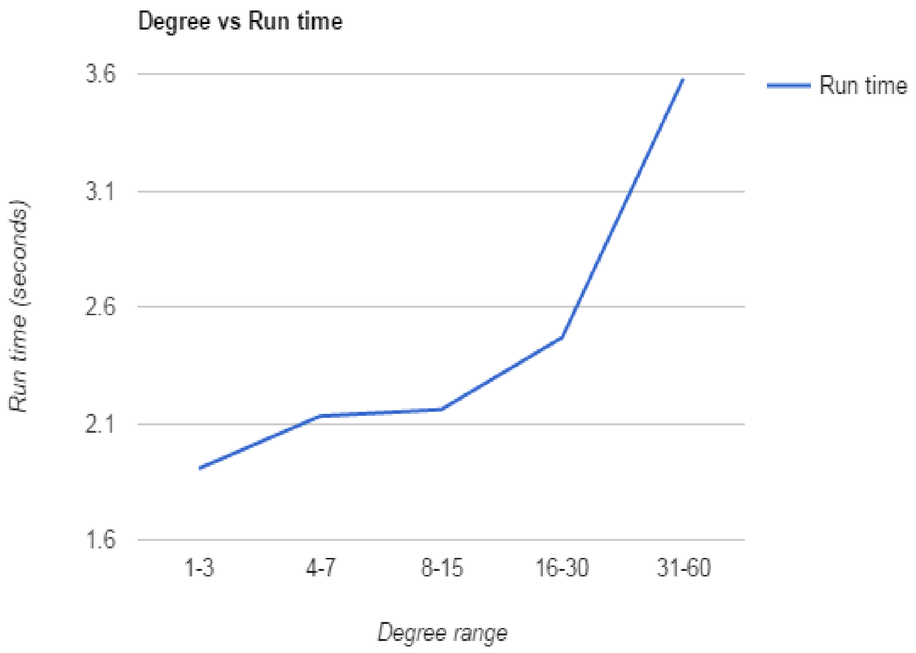


Figure 6.4: URL In-degree vs Run time

7 Conclusion

In this work, we formulate a graph-modeling problem for identifying unknown files and URLs with the help of the malware classification datasets at F-Secure Labs. We develop an iterative graph-based algorithm that can identify unknown downloader samples and download URLs. We evaluate the system using 4 months malware reputation data from F-Secure backend systems. Our experimental results show that the solution has effectively met the initial requirements that were defined during the problem definition. The system is able to automatically construct a graph model from the existing malware reputation systems, NRS and FRS, using OrientDB graph database. The system also meets the goal of identifying potential malware using graph traversal and mining techniques with 0% FPR. This solution functions in addition to the existing server-side malware classification methods and improves the current classification performance of malware at F-Secure by atleast 10%. The method is even able to identify few malware that are not currently identified by any of the anti-virus vendors in the market. Moreover, our solution is instrumental in exploring the infrastructures of some malware and thereby uncover the common patterns between malware. Hence, these results prove the accuracy and efficiency of our algorithm with real-world malware classification data.

7.1 Future Work

As future work, there are a number of improvements that can be made to the existing solution.

- Currently, the graph algorithm works the best in providing improved classification when there are enough malicious payloads in the input dataset. This can be improved by also traversing from unknown payloads to explore unknown download URLs and using additional parameters like WHOIS, number of URL hits to predict unknown URLs, in addition to their neighborhood.

- The input data to the current systems is from one of the many data source systems at F-Secure. Due to the query limitations of this system, the number of input samples for the experiments were limited for this study. But future work can combine multiple data sources to have a large input datasets to make better decisions.

- For this study, we only identify unknown downloaders and download URLs. However, this method can be combined with machine learning to design classifiers that can be trained to even predict the maliciousness of payloads when there is not enough relationship information available.

- The system can be improved by using complex graph analysis methods namely Page Rank, Random-Walk, etc. to extract more meaningful results from the graphs

A Appendix A

In this section, we list the malware terminologies for quick reference.

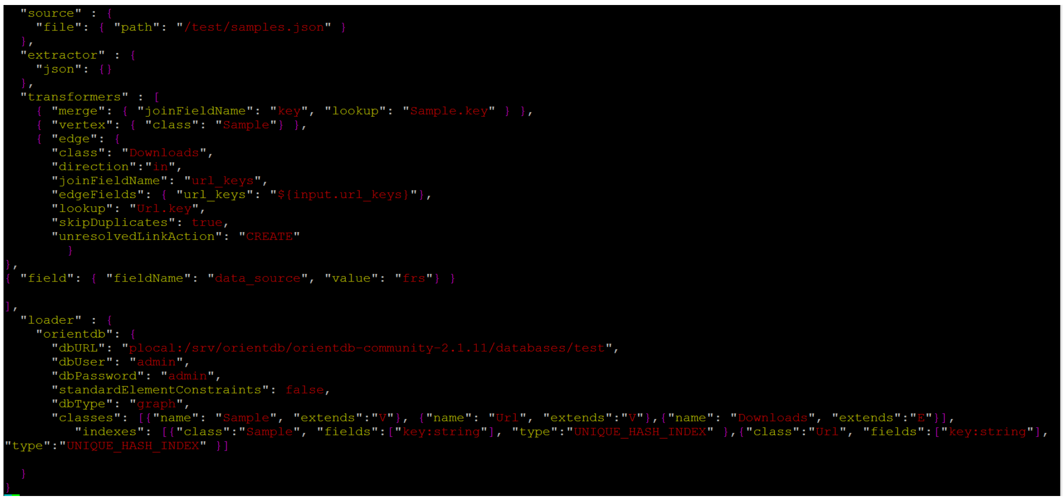
Term	**Synonyms**	**Description**
Malware	Malicious software, infected file.	Malicious programs which may include virus, worm, Trojan etc.
Reputation	Goodness or maliciousness, classification labels, detection.	A measure of goodness, can be used for URLs, files etc.
File sample	File, Sample, Software, executable, program, application.	A software instance, typically an executable file. Also mean word documents with embedded macros in this document.
URL	Webpage, URL link.	Uniform resource locator that represents a single web page document.
Downloader	Downloader sample, Downloader file.	A computer program, which aids in downloading supplementary files.
Payload	Payload sample. Payload file.	The eventual part of malware which performs a malicious action.
Malware Classification	Malware categorization, classification, detection.	The act of setting labels to unknown files and URls based on how they get executed, how they spread, and/or what they do.

Table A.1: Malware detection terminology

B Appendix B

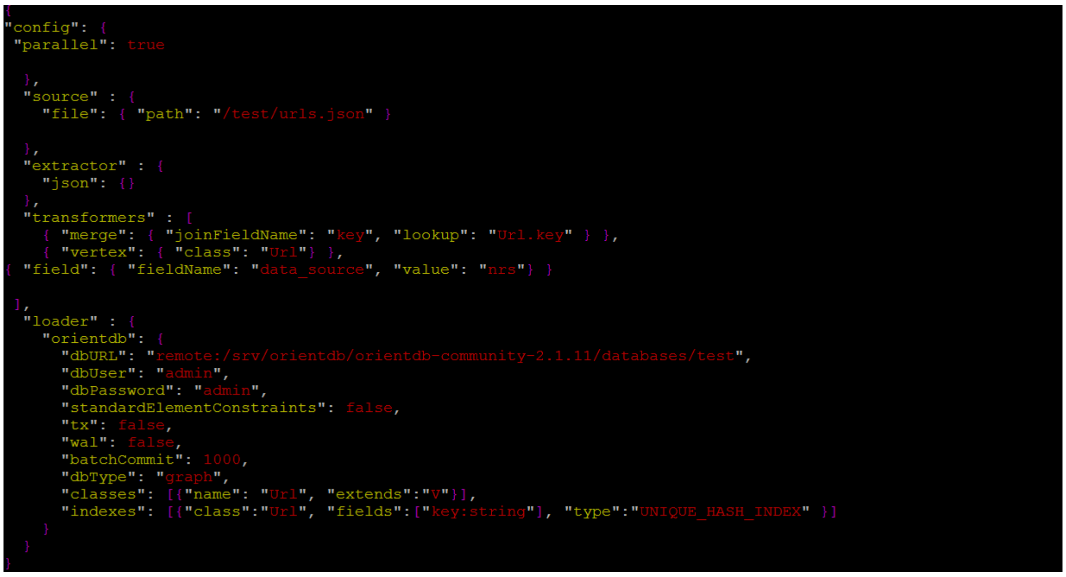
B.1 Graph Data Load Scripts

Figures B.1 and B.2 display the database load scripts used to load the datasets from the NRS and FRS systems onto the OrientDB Graph database. The vertex and edge definitions of the graph are specified in these scripts.



```
  "source" : {
    "file": { "path": "/test/samples.json" }
  },
  "extractor" : {
    "json": {}
  },
  "transformers" : [
    { "merge": { "joinFieldName": "key", "lookup": "Sample.key" } },
    { "vertex": { "class": "Sample"} },
    { "edge": {
      "class": "Downloads",
      "direction":"in",
      "joinFieldName": "url_keys",
      "edgeFields": { "url_keys": "${input.url_keys}"},
      "lookup": "Url.key",
      "skipDuplicates": true,
      "unresolvedLinkAction": "CREATE"
        }
},
{ "field": { "fieldName": "data_source", "value": "frs"} }

],
  "loader" : {
    "orientdb": {
      "dbURL": "plocal:/srv/orientdb/orientdb-community-2.1.11/databases/test",
      "dbUser": "admin",
      "dbPassword": "admin",
      "standardElementConstraints": false,
      "dbType": "graph",
      "classes": [{"name": "Sample", "extends":"V"}, {"name": "Url", "extends":"V"},{"name": "Downloads", "extends":"E"}],
        "indexes": [{"class":"Sample", "fields":["key:string"], "type":"UNIQUE_HASH_INDEX" },{"class":"Url", "fields":["key:string"],
"type":"UNIQUE_HASH_INDEX" }]

  }
}
```

Figure B.1: Load FRS Data



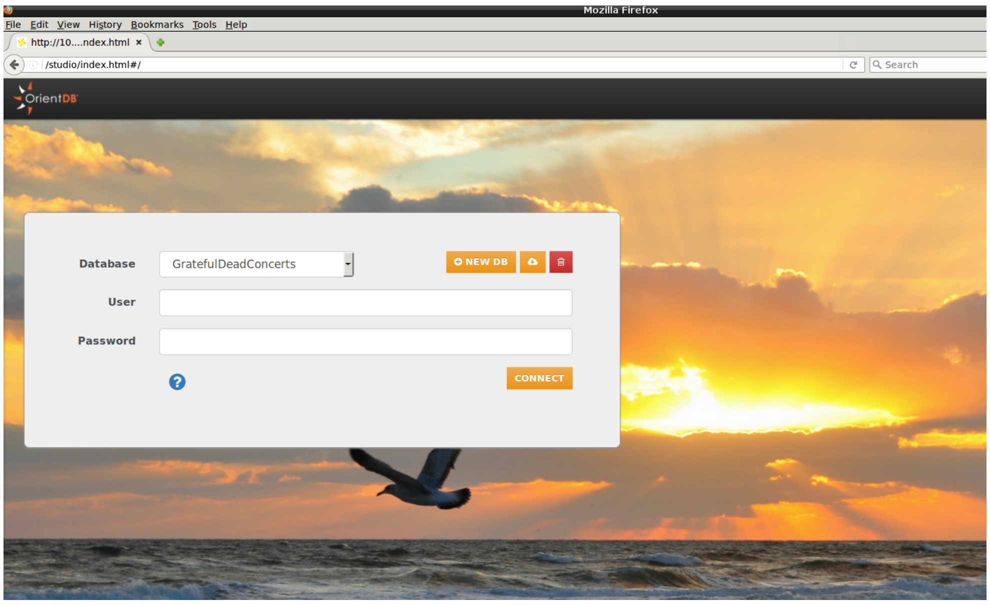
```
{
"config": {
 "parallel": true

  },
  "source" : {
    "file": { "path": "/test/urls.json" }

  },
  "extractor" : {
    "json": {}
  },
  "transformers" : [
    { "merge": { "joinFieldName": "key", "lookup": "Url.key" } },
    { "vertex": { "class": "Url"} },
{ "field": { "fieldName": "data_source", "value": "nrs"} }

 ],
  "loader" : {
    "orientdb": {
      "dbURL": "remote:/srv/orientdb/orientdb-community-2.1.11/databases/test",
      "dbUser": "admin",
      "dbPassword": "admin",
      "standardElementConstraints": false,
      "tx": false,
      "wal": false,
      "batchCommit": 1000,
      "dbType": "graph",
      "classes": [{"name": "Url", "extends":"V"}],
      "indexes": [{"class":"Url", "fields":["key:string"], "type":"UNIQUE_HASH_INDEX" }]
    }
  }
}
```

Figure B.2: Load NRS Data

C Appendix C

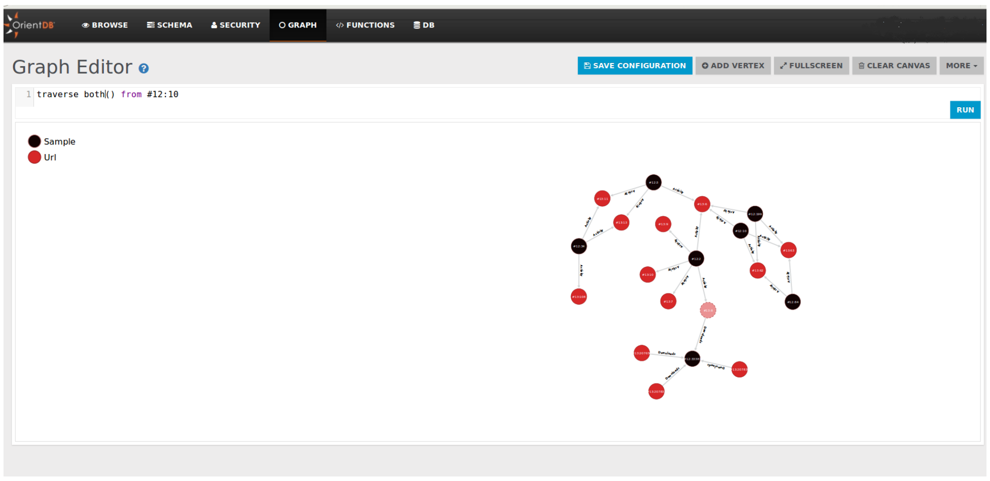
C.1 OrientDB Database Screenshots

Figures C.1 and C.2 show the visual editor of OrientDB for viewing the graph data and exploring patterns using traversal queries.



Database: GratefulDeadConcerts

Figure C.1: OrientDB Database



```
traverse both() from #12:10
```

Figure C.2: OrientDB Graph Editor

D Appendix D

D.1 Sample Graph Snapshots

Figures D.1 and D.2 display sample graph cluster snapshots from the test dataset.

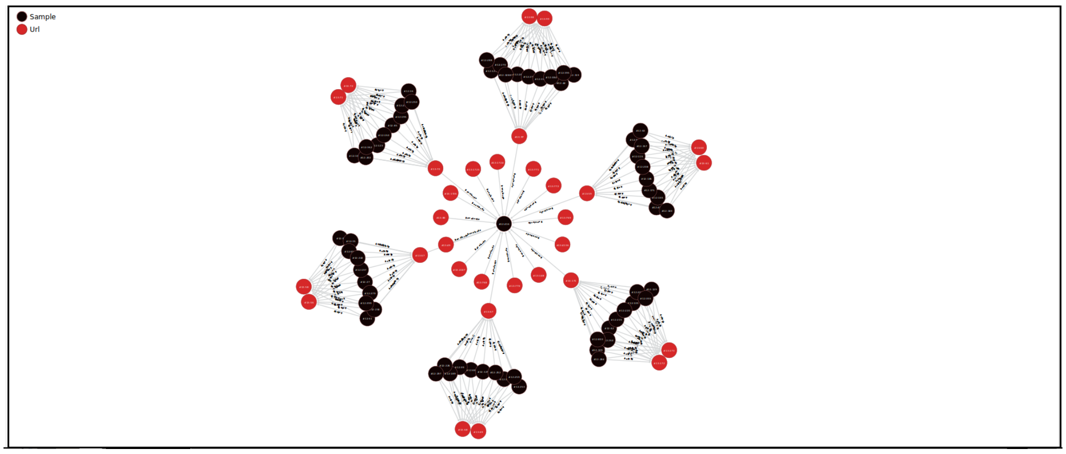


Figure D.1: Graph Snapshot 1

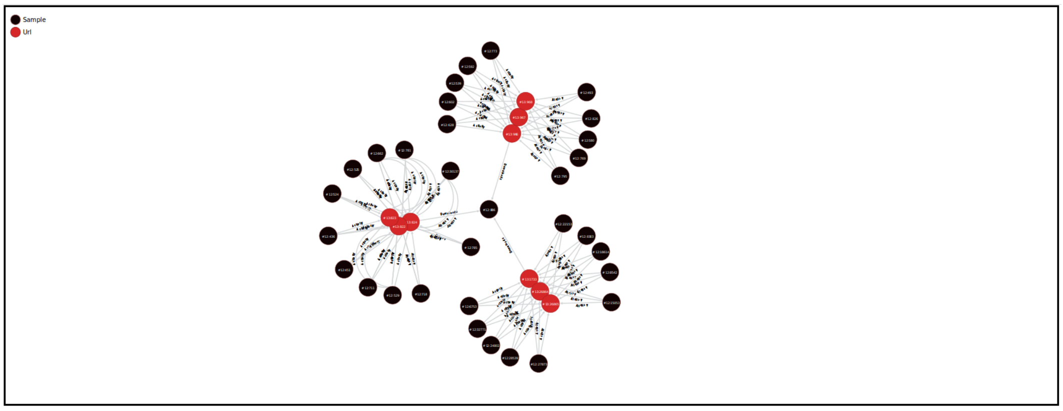


Figure D.2: Graph Snapshot 2

References

[1] M. Akiyama, T. Yagi, and M. Itoh. "Searching Structural Neighborhood of Malicious URLs to Improve Blacklisting". In: *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*. 2011, pp. 1–10. DOI: 10.1109/SAINT.2011.11.

[2] Alcatel-Lucent. *The Case for Network-based Malware Detection*. whitepaper: http://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2014/9599-case-network-based-malware-detection.pdf. 2014.

[3] Luca Becchetti et al. "Link Analysis for Web Spam Detection". In: *ACM Trans. Web* 2.1 (Mar. 2008), 2:1–2:42. ISSN: 1559-1131. DOI: 10.1145/1326561.1326563. URL: http://doi.acm.org/10.1145/1326561.1326563.

[4] Duen Horng Chau et al. "Polonium: Tera-Scale Graph Mining and Inference for Malware Detection". In: *SIAM INTERNATIONAL CONFERENCE ON DATA MINING (SDM)*. 2011, pp. 131–142.

[5] DAMBALLA. *Behind Today's Crimeware Installation Cycle: How Advanced Malware Morphs to Remain Stealthy and Persistent 10 (1st ed.* whitepaper: https://www.damballa.com/downloads/r_pubs/WP_Advanced_Malware_Install_LifeCycle.pdf. 2015.

[6] Son Dinh et al. "Spam campaign detection, analysis, and investigation". In: *Digital Investigation* 12, Supplement 1 (2015). {DFRWS} 2015 EuropeProceedings of the Second Annual {DFRWS} Europe, S12 –S21. ISSN: 1742-2876. DOI: http://dx.doi.org/10.1016/j.diin.2015.01.006. URL: http://www.sciencedirect.com/science/article/pii/S1742287615000079.

[7] F-Secure. *F-Secure Security Cloud*. whitepaper: https://www.f-secure.com/documents/996508/1030745/security_cloud.pdf. October 2015.

[8] F-Secure. *Threat Report 2015*. whitepaper: https://www.f-secure.com/documents/996508/1030743/Threat_Report_2015.pdf. 2015.

[9] Junxian Huang et al. "SocialWatch: Detection of Online Service Abuse via Large-scale Social Graphs". In: *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*. ASIA CCS '13. Hangzhou, China: ACM, 2013, pp. 143–148. ISBN: 978-1-4503-1767-2. DOI: 10.1145/2484313.2484330. URL: http://doi.acm.org/10.1145/2484313.2484330.

[10] Luca Invernizzi et al. "Nazca: Detecting Malware Distribution in Large-Scale Networks". In: (2014).

[11] Bum Jun Kwon et al. "The Dropper Effect: Insights into Malware Distribution with Downloader Graph Analytics". In: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA: ACM, 2015, pp. 1118–1129. URL: http://doi.acm.org/10.1145/2810103.2813724.

[12] Z. Li-xiong et al. "Malicious URL prediction based on community detection". In: *Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC), 2015 International Conference on.* 2015, pp. 1–7. DOI: 10.1109/SSIC.2015.7245681.

[13] M. Lindorfer et al. "ANDRUBIS – 1,000,000 Apps Later: A View on Current Android Malware Behaviors". In: *Proceedings of the Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS).* 2014, pp. 3–17.

[14] Juan Liu et al. "Graph Analysis for Detecting Fraud, Waste, and Abuse in Healthcare Data". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence.* AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 3912–3919. ISBN: 0-262-51129-0. URL: http://dl.acm.org/citation.cfm?id=2888116.2888258.

[15] Justin Ma et al. "Learning to Detect Malicious URLs". In: *ACM Trans. Intell. Syst. Technol.* 2.3 (May 2011), 30:1–30:24. ISSN: 2157-6904. DOI: 10.1145/1961189.1961202. URL: http://doi.acm.org/10.1145/1961189.1961202.

[16] *Malicious Software:A Security Threat to Internet Economy.* Ministerial Background Report: http://www.oecd.org/sti/40724457.pdf. 2008.

[17] Ashwin Athalye Nishant Doshi and Eric Chien. *Pay-Per-Install- The New Malware Distribution Network.* whitepaper: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/pay_per_install.pdf. 2010.

[18] Yin Minn Pa Pa et al. "IoTPOT: Analysing the Rise of IoT Compromises". In: *Proceedings of the 9th USENIX Conference on Offensive Technologies.* WOOT'15. Washington, D.C.: USENIX Association, 2015, pp. 9–9. URL: http://dl.acm.org/citation.cfm?id=2831211.2831220.

[19] Babak Rahbarinia, Marco Balduzzi, and Roberto Perdisci. "Real-Time Detection of Malware Downloads via Large-Scale URL-File-Machine Graph Mining". In: *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security.* ASIA CCS '16. Xi'an, China: ACM, 2016, pp. 783–794. ISBN: 978-1-4503-4233-9. DOI: 10.1145/2897845.2897918. URL: http://doi.acm.org/10.1145/2897845.2897918.

[20] Konrad Rieck et al. "Automatic Analysis of Malware Behavior Using Machine Learning". In: *J. Comput. Secur.* 19.4 (Dec. 2011), pp. 639–668. ISSN: 0926-227X. URL: http://dl.acm.org/citation.cfm?id=2011216.2011217.

[21] Bruce Snell. *Mobile Threat Report: What's on the Horizon for 2016.* whitepaper: http://www.mcafee.com/us/resources/reports/rp-mobile-threat-report-2016.pdf. 2016.

[22] V. S. Subrahmanian et al. "Types of Malware and Malware Distribution Strategies". In: *The Global Cyber-Vulnerability Report.* Cham: Springer International Publishing, 2015, pp. 33–46. URL: http://dx.doi.org/10.1007/978-3-319-25760-0_2.

[23] Andrei Venzhega, Polina Zhinalieva, and Nikolay Suboch. "Graph-based Malware Distributors Detection". In: *Proceedings of the 22Nd International Conference on World Wide Web*. WWW '13 Companion. Rio de Janeiro, Brazil: ACM, 2013, pp. 1141–1144. ISBN: 978-1-4503-2038-2. DOI: 10.1145/2487788.2488136. URL: http://doi.acm.org/10.1145/2487788.2488136.

[24] Sandeep Yadav et al. "Detecting Algorithmically Generated Malicious Domain Names". In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. IMC '10. Melbourne, Australia: ACM, 2010, pp. 48–61. ISBN: 978-1-4503-0483-2. DOI: 10.1145/1879141.1879148. URL: http://doi.acm.org/10.1145/1879141.1879148.

[25] Yanfang Ye et al. "Combining File Content and File Relations for Cloud Based Malware Detection". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '11. San Diego, California, USA: ACM, 2011, pp. 222–230. URL: http://doi.acm.org/10.1145/2020408.2020448.