A study of malicious software on the macOS operating system

Submitted in partial fulfilment
of the requirements of the degree of

Master of Science

of Rhodes University

Mark Alan Regensberg

Grahamstown, South Africa
January 2019

Abstract

Much of the published malware research begins with a common refrain: the cost, quantum and complexity of threats are increasing, and research and practice should prioritise efforts to automate and reduce times to detect and prevent malware, while improving the consistency of categories and taxonomies applied to modern malware. Existing work related to malware targeting Apple's macOS platform has not been spared this approach, although limited research has been conducted on the true nature of threats faced by users of the operating system. While macOS focused research available consistently notes an increase in macOS users, devices and ultimately in threats, an opportunity exists to understand the real nature of threats faced by macOS users and suggest potential avenues for future work. This research provides a view of the current state of macOS malware by analysing and exploring a dataset of malware detections on macOS endpoints captured over a period of eleven months by an anti-malware software vendor.

The dataset is augmented with malware information provided by the widely used VirusTotal service, as well as the application of prior automated malware categorisation work, *AVClass* to categorise and *SSDeep* to cluster and report on observed data. With Windows and Android platforms frequently in the spotlight as targets for highly disruptive malware like botnets, ransomware and cryptominers, research and intuition seem to suggest the threat of malware on this increasingly popular platform should be growing and evolving accordingly. Findings suggests that the direction and nature of growth and evolution may not be entirely as clear as industry reports suggest. Adware and Potentially Unwanted Applications (PUAs) make up the vast majority of the detected threats, with remote access trojans (RATs), ransomware and cryptocurrency miners comprising a relatively small proportion of the detected malware. This provides a number of avenues for potential future work to compare and contrast with research on other platforms, as well as identification of key factors that may influence its growth in the future.

Acknowledgements

First and foremost, significant thanks must be given to the anti-malware software vendor who wished to remain anonymous but gave of both their time and the data used in this research.

Secondly, the work would not have been possible with the support of the following people, places and things: Prof. Barry Irwin for creating safe spaces and triggers at the same time, the work of Patrick Wardle, Thomas Reed, Amit Serper and VirusTotal, without whom the macOS security community would be much worse off.

Finally, my family, who tried and generally failed to keep the noise down while I was trying to remember the word ‘telemetry’, and who have been unfailing in their support.

Dedicated to Finnian, a tireless defender of all things security related.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Hardware is easy to protect: lock it in a room, chain it to a desk, or buy a spare. Information poses more of a problem. It can exist in more than one place; be transported halfway across the planet in seconds; and be stolen without your knowledge.

Bruce Schneier, Protect Your Macintosh, 1994

Malicious Software, or as it is more popularly known, *malware*, has its origin alongside mass computing itself. The term *computer virus* was first discussed in detail by Fred Cohen, who noted *worms* and *trojan horses* in the same work as likely to become a significant threat to information security (Cohen, 1987). The first commonly accepted "infective" program was the *Creeper* software on the DEC PDP-10 in 1972 (Szor, 2005). At the time no more than non-malicious experimentation, it represented the conceptual genesis what would arguably become one of the larger families of threats faced by computer and network users in both personal and organisational contexts. *Elk Cloner*, released on the Apple II in 1982, was the first virus to resemble more modern counterparts by spreading relatively widely outside of its initial environment. The virus spread by infecting the boot sector of disks inserted on an infected machine, persisting in memory and triggering a payload after every 50th reset. While by all accounts distressing to those impacted by the virus, the author considered it little more than a prank and the practical impact was limited (Szor, 2005).

By definition, malware is software distributed and installed with malicious intent. Malicious software is not a new concept; creative terms like *Trojan Horses*, *Worms*, *Logic Bombs* and *Trap Doors* have described and categorised the tactics of software designed to exploit weakness and flaws in systems over the last five decades (Landwehr *et al.*, 1994; Soh *et al.*, 1995). The term "malware" is first referenced by Spafford (1994) in a continuation of his earlier work (Spafford, 1990) to represent the family of malicious software as a whole. Spafford also notes the appropriate use of *malware* to group software by its effect rather than its method. Thus, while malware can be seen as the progeny of earlier viruses, trojans and worms it also encompasses a broader definition: the entire group of malicious software than may impact a device, network or service with a specific focus on the malicious intent rather than the nature of propagation, payload or persistence.

A number of factors outside of the growth of personal computing have driven an increase in the spread of malware. The growth and ultimate ubiquity of the internet has provided a mechanism to automate and impact large volumes of user endpoints in an automated or semi-automated fashion (Provos *et al.*, 2007). Regardless of incentive, the vast network of connected devices has provided fertile ground for malware to evolve and adapt in response to increasing security awareness of users, organisations and malware analysis tooling that help develop detection and neutralisation techniques (for example, honeypots[1]) (Cabaj *et al.*, 2017). Along with this, the evolution of software engineering has in some ways assisted with this spread. As a maturing discipline, much of the tooling and building blocks of modern systems share dependencies (for example, lower level SSL libraries such as OpenSSL[2] or modern development frameworks such as Microsoft's ASP.NET MVC[3] and Ruby on Rails[4]), and a weakness in a single dependency can be used to spread malware over a greater surface area than flaws that may be present in a piece of software developed in relative isolation (Luszcz, 2017).

The increased diversity of connected devices has also played a role: the ubiquity of mobile devices (Zhou and Jiang, 2012) and Internet-of-Things (IoT) devices (Wang *et al.*, 2017) provide an ever increasing surface area, and result in a constantly evolving threat landscape. Finally, the creation of malware itself has increasingly become a commodity activity with the spread of malware creation toolkits (Ollmann, 2008; Moore *et al.*, 2009). The *Virus Construction Set* and *Genvir* were toolkits designed to simplify the creation of malware, available as early as 1990 (Szor, 2005). The frequently cited *Zeus* and *SpyEye*

[1]A security mechanism set to detect or deflect unauthorized use of information systems by emulating a valid system

[2]`https://www.openssl.org`

[3]`https://www.asp.net/mvc`

[4]`https://rubyonrails.org`

kits allowed for derivatives of a commonly used banking trojan to be built and spread by relatively unskilled individuals (Stevens and Jackson, 2010; Ye *et al.*, 2017). Similar toolkits are currently available for many common variants of malware by platform or attack vector, for example office document based exploits such as the *AKBuilder* exploit kit (Szappanos, 2016), trojan development kits for the Android mobile platform (Venkatesan, 2017), ransomware kits like *Tox* (McAfee, 2015) or Remote Access Tools (RATs) like *Poison Ivy* (FireEye, 2013) or the *DarkComet* RAT[5] (Breen, 2015).

Given the ubiquity, increasing threats and academic attention paid to malware (Ye *et al.*, 2017), the changing goals and underlying motivation for the creation of malware deserves mention. Why does malware exist to begin with? Economic incentives, socio-political factors and human-computer interaction all play a role in the malware ecosystem (Moore, 2010; Cavelty, 2018). While there is value in information and this may be a target for malicious software, more readily accessible value can be found in stored crypto-currencies or payment card information that malware may be designed to target (Moore *et al.*, 2009; Moore, 2010). More topical contemporary malware like ransomware (malware designed to lock access to system files or resources until a ransom has been paid) has clearly demonstrated the commercial motivation for malware designed to extort users (Kharraz *et al.*, 2015), while other disruption or denial of services made possible by malware enabled botnets may be used to extract revenge, extortion or prevent competition. Botnets comprising of large groups of malware infected machines make the identification of a single point of origin challenging (if not, in many cases, impossible) making attribution a commensurately more involved, multi-factor activity (Joshi and Pilli, 2016; Shamsi *et al.*, 2016).

Even legitimate business models can become adversarial through abuse: advertising malware, or adware, is malicious software that uses a variety of tactics to track and advertise to users leading to privacy and security issues as the lines between legitimate and malicious software blur (Urban *et al.*, 2018). Malware may also be spread to further specific political aims; nation states using malware to achieve political, economic and territorial goals have been observed and attributed (although rarely in absolute terms). Malware has been developed with precise targeting, as was suggested in the case of *Stuxnet* and the control systems in use in Iranian nuclear facilities (Marquis-Boire *et al.*, 2015; Rosenberg *et al.*, 2017), through to the broad, shotgun like approaches observed with *NotPetya* that impacted far more than the Ukranian targets thought to be the original primary objective (UK Foreign & Commonwealth Office, 2018).

[5]`https://archive.org/details/darkcometrat`

This mix of systemic factors, methods and motivations provides an increasingly complex backdrop and dynamic environment for contemporary malware research. In the sections that follow, the motivation, questions, objectives, outcomes and approach of this research will be discussed before reviewing the overall structure of the thesis that follows.

1.2 Motivation for this Research

The relevance and reasons for the study of malware are captured most effectively by Landwehr *et al.* (1994, p. 211): "knowing how systems have failed can help us build systems that resist failure". A thorough understanding of the tactics, motivation and spread of malware is therefore a key requirement to reducing the overall impact of malware in the future, both in an individual and organisational context.

While related work will be covered in detail in Section 2.1 onward, malware related research specific to Apple's macOS[6] operating system (previously known as Mac OS X between 2001 and 2012, and OS X between 2012 and 2016) has remained largely in the realm of specialist practitioners, with limited academic attention. The vast majority of malware related work found by this researcher has focused on Microsoft's Windows operating system and the Android mobile operating system. The use of macOS endpoints has, however, become more common. Enterprise management tooling for large fleets of macOS endpoints like JAMF[7] and Facebook's osquery[8] have seen rapid growth in recent years, spurred on by increasing number macOS endpoints requiring active management in the case of JAMF, and endpoint visibility in the case of osquery. From an information technology management perspective, Apple's operating system has become an increasingly noticeable part of business computing, and a largely unexplored area of risk.

If the perception of macOS as a more secure operating system is still common, it is increasingly difficult to justify. Before 2012, Apple themselves claimed that their operating system "doesn't get PC viruses. A Mac isn't susceptible to the thousands of viruses plaguing Windows-based computers" (Manning, 2012, p. 1). A number of examples of contemporary macOS malware have surfaced, yet there remains a gap in scholarly empirical research of malware on the platform.

[6]For consistency, macOS is used to refer to the operating system in this work, although other works referred to may use OS X or Mac OS X.

[7]`https://www.jamf.com`

[8]`https://osquery.io`

As a result, an opportunity exists to understand the real nature of threats faced by macOS users, and to build a foundation for further work that may look to understand how macOS malware is likely to develop in the future, how effective the current controls may be, and what factors are likely to influence this.

1.3 Research Questions and Objectives

This research hypothesises that malware is a credible threat to both individual and organisational users of the macOS operating system. Furthermore, by understanding the nature of that malware, the information can be practically applied to shape and improve the security posture of macOS users and administrators.

The research aims to test this hypothesis through the following research questions:

1. What malware families have been recently observed on the macOS platform?
2. How much diversity is there in the population of observed macOS malware?
3. Have detection rates increased over time?
4. What direction should future research take to understand the factors influencing malware on the macOS platform?

The overall objective of this research is to establish a view of the state of malware on the macOS platform. In order to do this, analysis of a real world dataset will be carried out, and the following secondary objectives addressed:

1. Aggregation and enrichment of the dataset using available resources and tools
2. Analysis of the aggregated and enriched data to identify the nature of information available
3. Classification and clustering of observed malware in the dataset by family, type and/or behaviour
4. Determine the overall nature and diversity of malware on the macOS platform using the analysed dataset

1.4 Expected Outcomes

In a study of a large dataset of malware detections on a network of Microsoft Windows endpoints, Yen *et al.* (2014) noted that the work had the capacity to inform security policy, security tooling and end user training. Similarly, the expected outcomes of this work include greater visibility into the real nature, and ultimately risk, of malware for users of the macOS operating system.

Additionally, other work in the field of malware research has provided a strong foundation on which to build with respect to classification and labelling of malware, clustering and understanding the diversity of malware datasets on other platforms. As a result, a contribution is expected as these works are applied to a malware dataset specific to the macOS platform.

Finally, as seen in work by Yen *et al.* (2014), Mezzour *et al.* (2017) and Urban *et al.* (2018), this work has relied on privileged access to real world datasets of malware encounters from security software vendors, and establishes a baseline for similar work on the topic in the future. As noted by Yen *et al.* (2014, p. 1118), such work contributes to "a portfolio of real-world research results that can illuminate broad practices and trends".

1.5 Approach

The data for this research was provided by the vendor of a desktop security product deployed in small, medium and large enterprise environments globally, who wished to remain anonymous. The vendor supplies anti-malware software for both the Microsoft Windows and macOS operating systems. The data provided for use in this research was limited to activity on macOS clients recorded between August 2017 and June 2018.

The approach taken required the extraction, processing and cleaning of dataset of observations. The data was imported into a database, and went through multiple stages of aggregation and enrichment using a mix of tools and scripts before analysis was conducted on the end result.

At a high level, the steps followed were:

1. Coalescing the data from the vendor provided files into a single file and importing selected data from the file into a relational database;

2. Construction of an aggregated dataset based on the initial import that contained records of the unique samples observed;
3. Enrichment of the aggregated dataset with external data sources;
4. Categorisation and classification of the observed records; and
5. Exploratory analysis of the datasets both individually and as a fused dataset.

1.6 Document Structure

Following on from the introduction, context and description of this research, the remainder of the thesis is structured as follows:

- Chapter 2 provides background to the research area along with related work on malware detection and analysis, classification and labelling and the macOS operating system.
- Chapter 3 reviews the methodology followed, with details on the approach taken, experimental setup, steps followed to process the data and validation.
- Chapter 4 presents the results of analysis of the resulting datasets, contrasting with results from similar works on other operating systems where possible.
- Chapter 5 discusses the findings from the analysis, the research questions posed, limitations of the research and potential future research opportunities.
- Chapter 6 summarises the work and concludes the thesis.

Chapter 2

Background and Related Work

No matter who you are, no matter where you live, and no matter how many people are chasing you, what you don't read is often as important as what you do read.

Lemony Snicket

2.1 Introduction

The nature and behaviour of malware is a multifaceted topic with a long research history. As with any research discipline, certain areas attract more attention that others; contemporary issues in organisational information security, ongoing improvements to data privacy regulations, the international political landscape and significant media coverage and awareness of ransomware outbreaks have continued to raise the profile of malware globally in both practice and academic circles.

The related work in this section is broken into four further subsections. Section 2.2 considers the key areas of malware study, specifically identification of malware, methods of detection and tooling and the analysis of malware. Section 2.3 examines aspects of infections, including common terminology used to describe malware, how malware infects and remains on the host once infected, and the role of operating system controls such as code signing play in preventing malware. Section 2.4 reviews work related to the naming and classification of malware and malware related taxonomies. Finally, Section 2.5 looks at work related specifically to macOS, including a synopsis of the differences in malware execution, persistence, detection and behaviour between the macOS platform and other, more commonly studied operating systems will be considered.

2.2 Understanding Malware

Neither malware research nor protection could take place without the ability to detect, analyse and understand the activities of malware. In the sections that follow, work on detection and analysis of malware will be discussed as well as work on the ongoing evolution of malware.

2.2.1 Detection and Analysis

Effective analysis of malware is key to detecting and defending against it. Kolbitsch *et al.* (2009, p. 352) note that "it is crucial that the insight obtained through malware analysis is translated into detection and mitigation capabilities that allow one to eliminate malicious code running on infected machines". Anti-malware software (traditionally referred as anti-virus software) is often the primary defence of many computer users against malware, and an ever-present part of organisational malware threat management.

Analysis of malware can be performed statically or dynamically. Static analysis is analysis carried out without executing a file or program, extracting information either through disassembly or evaluation of non-binary data contained within a file. Dynamic analysis makes use of program execution, often in a virtual or sandbox environment, to monitor system or API calls, registry or file changes, memory activity or network activity (Szor, 2005; Sikorski and Honig, 2012). Both methods have advantages and disadvantages: static analysis is faster and less computationally expensive compared to dynamic analysis while dynamic analysis provides a richer set of underlying heuristics for analysis as actual behaviour can be tracked (Kolbitsch *et al.*, 2009; Damodaran *et al.*, 2017). In order to evade detection or make static analysis more difficult, malware authors can apply a number of counter measures to ensure known signatures are more difficult to detect (Sikorski and Honig, 2012). This could include packing the file (effectively compressing and obfuscating the bytecode), encrypting it, recompiling it with minor changes or unused execution paths (also changing the compiled bytecode) or altering behaviour dynamically (polymorphic or metamorphic malware) in order to mutate the underlying malware sufficiently to avoid detection (Szor, 2005; You and Yim, 2010; Sikorski and Honig, 2012; Baysa *et al.*, 2013;

Ye *et al.*, 2017). These evasion techniques are most effective against static analysis, as dynamic analysis monitors actual behaviour. However, dynamic analysis is not a completely immune to morphic techniques; malware may detect operation in a sandbox or debugger and alter behaviour accordingly to counter dynamic analysis techniques (Egele *et al.*, 2012), although Damodaran *et al.* (2017) found that this may not occur often enough in practice to make a marked difference to larger scale categorisation effors using dynamic analysis.

Using both static and dynamic analysis, contemporary research has focused on three principle methods of detecting malware: signature based detection, behaviour based detection and statistical based detection (Damodaran *et al.*, 2017). The first method, signature based detection, has been and continues to be the mainstay for malware detection in the anti-virus industry. At a fundamental level, signature matching is matching a sequence of bytes in a file to known malware (Damodaran *et al.*, 2017). Signature matching is fast, requires relatively little interpretation and has proven to be the foundational method of malware detection. However, it is also the method most prone to evasion by morphic behaviour, and requires an up to date database of signatures to be available in order to be effective: new strains of malware require new signatures, and signatures may often take some time to become available (Moser *et al.*, 2007), a key limitation of this method of detection. Researchers have attempted novel ways of reducing signature databases (Nachenberg and Griffin, 2012), or attempting to work with smaller, more generic signatures to defeat counter measures (Poston, 2013). In an analysis of one anti-virus product making use of the latter approach, Ormandy (2011) demonstrated that often these attempts carry more marketing value than practical advantage. The challenges and potential inefficiency of signature based detection methods are also noted by Moser *et al.* (2007) and Kolbitsch *et al.* (2009).

In the second method Jacob *et al.* (2008) suggested that behavioural based detection could use both static and dynamic analysis to establish baseline patterns for regular, non-malicious behaviour and thus detect malware through anomalous behaviour. Adapting strategies similar to intrusion detection systems, the authors also make note of work that uses Markov models to establish a meaningful statistical inference of valid behaviour, as seen in Zanero (2004). Following on from this use of statistical inference, statistical based detection has extended the use of statistical models to determine malicious behaviour. At it's simplest, statistical based detection may make use of both static and dynamic analysis to determine models of behaviour based on an observed set of behaviours, features or interactions. The use of Hidden Markov Models (HMM), a machine learning technique common in many pattern matching applications like handwriting or speech recognition, have been used to great effect in malware detection (Damodaran *et al.*, 2017). HMMs allow for a discrete set of observables (in the case of malware this could be opcodes, instructions, API calls, or any other observable pattern) derived from static or dynamic analysis to be used as training data and in turn determine the probability of a given file being malware. The technique has been applied to static analysis (Annachhatre *et al.*, 2015), dynamic analysis (Dai *et al.*, 2009; Kolbitsch *et al.*, 2009) and a combination of the two (Damodaran *et al.*, 2017). Other related statistical methods used have included the use of n-grams and methods of clustering (Wong and Stamp, 2006; Kolter and Maloof, 2006; Raff *et al.*, 2016; Pai *et al.*, 2017). Finally, it should be noted that detection may occur outside of the host environment through external network analyses, including logitudinal and meta-analysis of external behaviours, as seen in Stalmans and Irwin (2011), Irwin (2013) or Lever *et al.* (2017).

2.2.2 Evolution of Malware

Understanding the evolution of malware plays a crucial role in classification, establishing well reasoned taxonomies and determining trends. From the outset malware research has frequently borrowed both structure and etymology from the biological sciences (Spafford *et al.*, 1991; Spafford, 1990). This approach is not without limits; in many ways the evolution of malicious software is simplistic compared to the complex interactions, competition for resources and much longer evolutionary timeframes observed in nature (Seideman *et al.*, 2015a). However, biological approaches are used successfully in understanding the evolution of malware. Karim *et al.* (2005, p. 13) note:

> “Malware authors use generators, incorporate libraries, and borrow code from others. There exists a robust network for exchange, and some malware authors take time to read and understand prior approaches. Malware also frequently evolves due to rapid modify-and-release cycles, creating numerous strains of a common form. The result of this reuse is a tangled network of derivation relationships between malicious programs.”

Given the use of shared code, multiple derivatives and related behaviours, the evolution of malware can therefore benefit from phylogenetic modelling to better understand similarities, trends and evolution of malware over time (Karim *et al.*, 2005). Proposed by Karim *et al.* (2005), the use of phylogeny models provides a tree-based model to show relationships between entities (in this case, malware samples) with similarity determined by comparing n-grams (order sensitive n-length byte strings) or n-perms (order insensitive n-length byte strings) between samples to determine a likely evolutionary relationship based on computed closeness. The branching diagram made possible by phylogeny models allow for parent-child relationships to be determined, with a leaf nodes able to have multiple parents as well as abstract nodes where no parents have been established, as would be seen in the case of completely new malware with no shared evolution. Similar work by Gupta *et al.* (2009) used a graph pruning algorithm (i.e., edges within the malware family graph are fully connected, and pruned to decompose and demonstrate likely relationships) to identify the evolution of 669 distinct families of malware from a longitudinal dataset gathered by an anti-virus company over a nine year period. The authors used analysis of malware metadata text as the basis of comparison rather than data derived from static or dynamic analysis.

Building on the earlier work by Karim *et al.* (2005) using phylogenetic models as well as classification work using n-grams by Kolter and Maloof (2006), Seideman *et al.* (2015a) mapped evolutionary relationships and biodiversity using similarity of features extracted from static analysis of a sample set. This approach has two shortcomings: the first is the previously discussed challenge of code packing and obfuscation that static analysis is subject to, and the second is the difficulty in representing ‘reticulation events’, or re-use of shared code fragments between malware samples (Liu *et al.*, 2016). Seideman *et al.* (2015b), Liu *et al.* (2016) and Moubarak *et al.* (2017) have used phylogenetic systematics to demonstrate evolutionary relationships through dynamic or behavioural analysis. The approach has also been used to group and understand evolutionary relationships in remotely executed vulnerabilities by Ma *et al.* (2006).

Not all studies of malware evolution make use of biological terms of reference. Prior work done within the Android malware ecosystem by Zhou and Jiang (2012) considered the analysed corpus in terms of behaviour, detection and the overall discovery timeline, although evolutionary considerations were limited to specific samples within the overall sample set. In addition, mobile platforms have provided opportunities to explore the relationship between evolution and the relative popularity of a platform. Suarez-Tangil *et al.* (2014) considered both market share as well as the platform security model in their survey of the evolution of mobile device malware, focusing on the bahaviour of malware, infection vectors and mechanisms employed to escalate privilege to categorise samples. Rodríguez (2017) analysed malware targeting point of sale (POS) systems, focusing on functionality, methods of persistence and data exfiltration over a six year period, again with a reasonably small sample group and on a niche platform. Calleja *et al.* (2016)

took the novel approach of studying forty years of malware evolution through traditional software engineering metrics, including codebase size, estimated development cost and code quality.

Broadly, a review of the literature appears to show studies considering evolution fall into two main approaches. The first approach considers metrics derived directly from instruction sets, often from static analysis, used to determine similarities, common sequences and differences. These studies appear to benefit from larger data sets, especially those with a longer temporal base. The second uses behaviour, ranging from API calls to activities observed in dynamic analysis to build up an understanding of potential relationships across a timeline. These may be more suited to smaller datasets, often in more focused ecosystems. Both may have have advantages and disadvantages depending on the end goal and operating environment, and both present prior work relevant to this research. Studies following either approach have made beneficial use of phylogeny models to represent evolutionary relationships. In the section that follows related work on classification and taxonomies for malware will be considered.

2.3 Observing Malware in the Wild

Real world observations of malware can be described from a number of standpoints. Terminology used to describe malware ranges from the evergreen ‘computer virus’ to newer descriptions of ransomware and cryptominers. As these terms are used throughout this research, a review of common terminology is included. Following this, methods of infection and persistence are reviewed, as well as a review of code signing in malware.

2.3.1 Common Terminology

As discussed in Section 1.1, computer viruses have long been grouped by their method of infection and payload. Over time these characteristics change, however a common high level taxonomic terminology remains. Efforts to create detailed and consistent taxonomies for use in research are discussed in Section 2.4.2, however a description of the commonly used terminology to describe high level categories is justified.

In his seminal work, Szor (2005) defines a number types of malicious programs. Many of the types mentioned have fallen from common use, for example *octopuses* (a conceptual precursor to the modern botnet), *rabbits* (a single instance virus) and *logic bombs* (malicious logic written into legitimate applications). Other types have fallen from use as the technology landscape has changed as seen in the case of *diallers*[1], a technology and terminology no longer in frequent use. A number of terms both from the work in question and others are still in common use, and are shown in Table 2.1 along with the description and source.

Of particular interest in this research are adware and Potentially Unwanted Programs (PUPs). Often grouped along with spyware, adware and PUPs represent a family of malware occupying murky territory in terms of overall legitimacy, a common descriptive thread being deceptive behaviour and non-consensual activity. McFedries (2005, p. 72) provides the following context:

> Linguistic proof of the cultural impact of spyware is the large number of synonyms that have popped up in the past year or so. These include sneakware, stealthware, snoopware, trackware, thiefware, or, tellingly, scumware. A spyware program is also sometimes called an E.T. application, because it phones home to secretly send data to an online destination.

Geniola *et al.* (2017) considers the role of download portals and software aggregation sites in spreading adware and PUPs, noting both privacy and security issues with the spread of this group of malware. Kotzias *et al.* (2015) examines the prevalence of abuse of code signing by PUPs, likely an artefact of attempted legitimacy and a topic discussed in greater detail in Section 2.3.3.

Other types of malware noted as relevant contemporary threats can be found in industry reports such as Symantec (2018a) and Kaspersky (2018), which call out ransomware and coinmining as the fastest growing threats for the period under investigation. In addition to this the reports highlight a variety of malware variants that defy simple categorisation, but can broadly be seen as backdoors, trojans and RATs in that they infect a users machines through a variety of mechanisms, and execute additional payloads depending on the target, operator and vulnerabilities available to exploit. An example of this is the *emotet* banking trojan and dropper (Symantec, 2017), which had a significant detection rate world wide (Symantec, 2018a). The various categories of malware relevant to this research will be discussed in more detail in Section 3.8.

[1]A dialler is malware installed to automatically dial numbers for malicious, usually revenue generating, purposes (Szor, 2005).

Table 2.1: Malware terminology

Term	Definition	Source
Adware	Malware that displays advertisements or promotion without the users knowledge or consent	Szor (2005)
Backdoor	Malware that allows remote connections to a system without the users consent	Szor (2005)
C2	Command and Control, provides remote instructions to locally persisted malware	Gu *et al.* (2008)
Cryptominer	Malware that uses computer resources for cryptocurrency mining (also known as coinmining)	Tahir *et al.* (2017)
Dropper	Malware used to download additional malware or instructions, either immediately or at a later date based on instruction from a C2	Kwon *et al.* (2015)
Keylogger	Malware that captures a users keystrokes, including potentially sensitive information	Szor (2005)
PUP	Potentually Unwanted Program: any program installed without a users consent for an undisclosed purpose	McFedries (2005)
Ransomware	Malware that encrypts stored files, charging a fee (ransom) to provide the decryption key	Gazet (2010)
RAT	Remote Access Tool: provides remote connectivity and access to local devices, potentially including storage and media devices	Alperovitch (2011)
Rootkit	Software that allows privileged access to a computer without being detected, often as part of other software	Szor (2005)
Spyware	Malware that spies on a users behaviour, usage and activity	Szor (2005)
Trojan	Malware included in a different program or application, installed unknowingly by a user	Szor (2005)
Virus	Code that replicates a potentially modified version of itself	Szor (2005)
Worm	A virus that replicates over network resources	Szor (2005)

2.3.2 Methods of infection and Persistence

Like a biological virus, all malware requires a way of infecting the host. Also known as the infection vector, hosts can be infected using direct or indirect means. Examples of direct mechanisms of infection may include physical device infection through insertion of a USB drive (Pham *et al.*, 2010), untrusted accessories (Lau *et al.*, 2013; Zdziarski, 2014) or network vulnerability like the SQL Slammer worm (Moore *et al.*, 2003) or *Wannacry* ransomware (US-CERT, 2017). Indirect infection vectors are infection methods that rely on user action of some sort, like visiting an infected website, downloading an infected application or opening an email attachment with a malicious attachment (Hang *et al.*, 2016; Mezzour *et al.*, 2017). Simmons *et al.* (2009) provide a robust review of infection vectors as part of the AVOIDIT (Attack Vector, Operational Impact, Defense, Information Impact, and Target) taxonomy.

Once a successful initial infection has taken place, malware requires a method of persistence in order to be effective. Persistence methods vary between operating system tand hardware platforms, usually relying on similar methods of persistence to legitimate applications installed on the host. Most relevant to this research are the persistence mechanisms summarised by Wardle (2014), and listed in Table 2.2.

2.3.3 Abuse of trust: malware and code signing

Determining which applications, installers and executables are trustworthy is an important operating system control. The ability to validate a file, be it an installer or an already installed executable, has been provided by both mobile and non-mobile operating systems through code signing (Kim *et al.*, 2017). The process of certification relies on the use of Public Key Cryptography (PKC) to determine both the authenticity and integrity of a file. In the context of code signing the method suffers from systemic rather than technical weaknesses: Kim *et al.* (2017) found that inadequate client side signature validation, compromised publisher keys and key mismanagement, and the failure of Certificate Authorities (CA) to verify potentially malicious actors were key issues in code signing efficacy on the Microsoft Windows operating system. Alrawi and Mohaisen (2016) provide exploratory research of a signed malware corpus, noting significant similarities in trust chain length (the number of certificates in the certificate store of a digital signature), and issue, expiry and validity periods.

Table 2.2: macOS malware persistence mechanisms adapted from Wardle (2014)

Mechanism	Description
Low Level or Firmware	Code may be installed within the pre-boot load area, i.e. before the OS has booted
Kernel Extensions	Kernels extensions run at the highest OS privilege level. Ideal for rootkits.
Launch Agents	Most common user mode persistence mechanism, run before user login on startup
Cron Jobs	A common UNIX and Linux mechanism for running scripts or executables at regular intervals
Rc.common	Infrequently used macOS startup script common to BSD UNIX derivatives
Login/Logout Hooks	Technically deprecated script hook run on login or logout
Login Items & Sandboxed Login Items	Executed on login. Technically deprecated, but replaced with sandboxed login items
Re-opened Applications	Application state is maintained by the operating on system restart, can be abused by malware
Startup Items	Applications loaded on startup, visible to the user as startup items in system preferences
Launchd.conf	Companion configuration to launch agents, which can trigger script execution in its own right
Arbitrary dynamic libraries (DYLD load)	The DYLD_INSERT_LIBRARIES environment variable specifies arbitrary libraries that may be loaded alongside a process
Executable (MACH-O) injection	Unsigned binaries may allow arbitrary code to be injected and run by modifying the executable entry point
Application specific persistence	Applications that support plug-ins (for example, browsers) may have the functionality subverted to load malware

In both the Windows and Apple ecosystems, the barrier to obtaining a code signing certificate is in part financial, with fees that need to be paid, and in part validation by either third party CAs[2], or in the case of Apple, following a validation process by Apple themselves[3]. Analysis of certification of adware, PUPs and pay-per-install software by both Kotzias *et al.* (2015) and Serper (2018) show that neither the commercial nor validation efforts appear to be completely effective. Additionally, from a technical perspective the enforcement of code signing on macOS has been found to have technical flaws in implementation, allowing it to be bypassed (Levin, 2015; Reed, 2018a).

2.4 Malware in Research

Historic categories of malware receive limited useful attention in current literature. As discussed previously, Szor (2005) provided a high level taxonomy of malware types that, while frequently cited for context, have become less relevant in contemporary research. Viruses, worms, mass-mailers, octopuses, rabbits, germs, dialers, backdoors, trojans, rootkits, adware and keyloggers may all exist, but arguably represent more an infection vector than a discrete malware type. Contemporary malware authors have adapted to changes in operating systems, available vulnerabilities and more sophisticated financial and political incentives. For example, modern Advanced Persistent Threats (APT) may use multiple vectors to gain entry, maintain persistence and ultimately achieve their goals (Chen *et al.*, 2014; Symantec, 2018a). This rapid growth of malware techniques and platforms may have lead to earlier, infection vector-centric categorisation becoming outdated, but has not removed the value or challenge of consistent categorisation or relevant taxonomies.

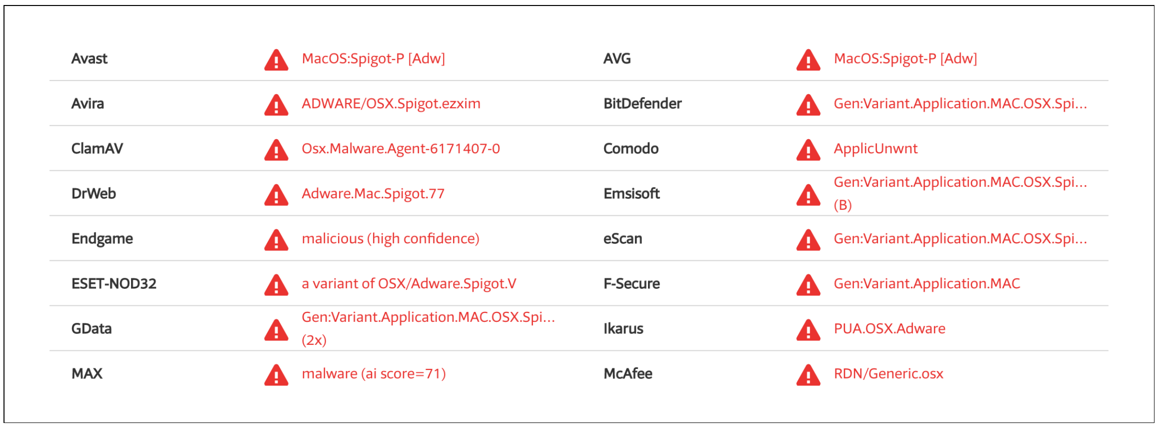
A distinction must be made between malware naming, labelling, categorisation and taxonomies. Specifically, related work will be considered in two discrete areas: naming and labelling of malware in Section 2.4.1, and categorisation and taxonomies in Section 2.4.2. Naming and labelling frequently adopts a more technical approach with significant efforts in automated labelling, and categories and taxonomies are often considered from a more theoretical standpoint. In the sections that follow, both will be considered in turn.

2.4.1 Naming and Labelling

Jackson (1990) identified the naming of malicious software as a challenge similar to that

[2]`https://docs.microsoft.com/en-us/windows/desktop/appxpkg/appx-portal`
[3]`https://developer.apple.com/support/certificates/`

faced by other branches of science at a time when a total of 46 different viruses were known for the PC (excluding the Macintosh operating system). Even at this early stage, the author noted the importance of a naming consistency within the virus research community arguing that discovery should be treated similarly to that of new astronomical or biological discoveries: immediately logged and categorised by some central authority. The principle challenge to this approach has been in part the role of anti-virus software vendors, for whom naming was simply a side-effect of detection, and the rapid increase over time in the volume of malware observed. From the 46 viruses catalogued in 1990, Kantchelian *et al.* (2015) reported a single malware vendor receiving approximately 344,000 unique instances in a day in 2014. This high volume makes manual naming a prohibitively expensive task, further complicated by a lack of agreed and adopted standard for naming malware across the information security industry and between vendors of anti-malware software (Kantchelian *et al.*, 2015; Hurier *et al.*, 2016, 2017). An example of disparate vendor labelling can be seen in Figure 2.1 taken from the VirusTotal service[4], with a single sample of the *spigot* macOS malware identified with a number of labels by different vendors.



Avast	MacOS:Spigot-P [Adw]	AVG	MacOS:Spigot-P [Adw]
Avira	ADWARE/OSX.Spigot.ezxim	BitDefender	Gen:Variant.Application.MAC.OSX.Spi...
ClamAV	Osx.Malware.Agent-6171407-0	Comodo	ApplicUnwnt
DrWeb	Adware.Mac.Spigot.77	Emsisoft	Gen:Variant.Application.MAC.OSX.Spi... (B)
Endgame	malicious (high confidence)	eScan	Gen:Variant.Application.MAC.OSX.Spi...
ESET-NOD32	a variant of OSX/Adware.Spigot.V	F-Secure	Gen:Variant.Application.MAC
GData	Gen:Variant.Application.MAC.OSX.Spi... (2x)	Ikarus	PUA.OSX.Adware
MAX	malware (ai score=71)	McAfee	RDN/Generic.osx

Figure 2.1: Various vendor labels for a Spigot malware sample, from VirusTotal

A number of attempts have been made to formalise conventions for malware naming and categorisation. The Computer Antivirus Research Organisation (CARO) was established in 1991, releasing the first formal naming convention for virus research (Skulason *et al.*, 1991). While the effort did not see long term adoption, it played a role in advancing thought around malware categorisation and highlighted the challenges faced with naming and categorisation within the rapidly growing and diverse population of malware (Scheidl, 1999; Riau, 2002; Szor, 2005). Following on from CARO and building on the success of the Common Vulnerability and Exposures[5] (CVE) initiative, the Common Malware Enumer-

[4]`https://virustotal.com`
[5]`https://cve.mitre.org`

ation[6] (CME) was developed to assist with universally accepted malware identification in a similar way that CVE's identified vulnerabilities. The platform was ultimately retired as the categorisation became less appropriate for threats being observed:

> "In late 2006 the malware threat changed away from the pandemic, widespread threats CME was developed to address to more localized, targeted threats, which significantly reduced the need for common malware identifiers to mitigate user confusion in the general public." (MITRE, 2006, p. 1)

As malware has evolved, so more recent thinking has shifted away from identification of malware in purely biological or tactical terms like virus, trojan or backdoor. Concepts of evolution and shared roots notwithstanding, the identification of malware has itself evolved to include the threat it represents rather than an isolated view of programmatic attributes. In other words, while biological constructs like phylogeny may explain much of the historical and development path of malware, when viewed from a threat perspective malware may be more easily categorised through a composite view of observed attributes: behaviours, family, infection sources and observable artefacts (for example, files, network connections or processes). The Malware Attribute Enumeration and Characterization[7] (MAEC) project is a community driven initiative and third major attempt at malware categorisation and naming building on the previous two. MAEC is a language to describe malware behaviour patterns and resultant threats (MITRE, 2018), allowing a close interactions with threat sharing platforms like the Malware Information Sharing Platform[8] (MISP).

Outside of threat sharing, practical solutions to the challenge of consistent naming for research purposes has found some solution in the large malware datasets now available to researchers, like VirusTotal. The advantage of these (and similar) datasets is that labelling from multiple anti-malware software vendors is available allowing for researchers to build more effective methods of consistent labelling for reporting. *AVClass* is an automated solution proposed by Sebastián *et al.* (2016) that can processes high volumes of vendor labels, resulting in a normalised, alias aware set of labels for a given malware dataset. Hurier *et al.* (2017) also considered the question of mislabelling and the relative 'noise' in categorisation found between different anti-malware vendors, proposing a similar approach for the processing a large datasets of Android malware.

[6]`http://cme.mitre.org`
[7]`http://maecproject.github.io`
[8]`http://www.misp-project.org`

While processing of vendor labels has provided an efficient mechanism for large scale labelling of malware datasets, it is not without its challenges. Bailey *et al.* (2007, p. 178) noted “that different AV products characterize malware in ways that are inconsistent across AV products, incomplete across malware, and that fail to be concise in their semantics”, and proposed a solution was based on clustering similar malware rather than standardisation of labelling. Kornblum (2006) proposed and implemented a method for using context triggered piecewise hashes to cluster and identify similar malware samples, based on earlier work on unsolicited email (SPAM) and file synchronisation by Tridgell (1999). The result of this work, the *SSDeep* tool[9], has become a standard way to efficiently cluster malware.

Finally, one of the advantages of homogeneous labelling over a large dataset of observations is the ability to determine diversity of the observed population. The Shannon diversity index is an index commonly used to indicate the diversity of species in population (Shannon, 1948), used by Soto-Valero and González (2018) as a measure of diversity of malware population. The formula for calculating the Shannon diversity index is:

$$-\sum_{i=1}^{n} p_i \; ln \; p_i \tag{2.1}$$

where the proportion of a malware families i relative to the total number of families (p_i) is multiplied by the natural logarithm of this proportion(lnp_i), and the result multiplied by -1.

2.4.2 Categorisation and Taxonomies

Im and Baskerville (2005, p. 69) define taxonomies as “the theory and practice of classification, arising in a branch of science known as systematics. Unlike nomological science with its focus on uniformity, the taxonomies of systematics focus on diversity”. Malware research is no stranger to the use of taxonomies for categorisation. Early work by Weaver *et al.* (2003), Karresand (2003), Szor (2005) and Filiol (2006) propose and discuss taxonomies representing the relative spread of malware at the time, with almost all acknowledging the rapid growth in tactics and behaviours. A taxonomy of botnet structures is proposed by Dagon *et al.* (2007) and extended to include other attributes by Khattak *et al.* (2014).

[9]`https://ssdeep-project.github.io/ssdeep/index.html`

Grégio *et al.* (2015) propose a comprehensive taxonomy of behaviours, noting that while the original classes of malware are suitable for simple categorisation, "classes may intersect their expected behaviors" (p. 2759) thus making the clear distinction between simplistic classes of malware grouped by their expected behaviour and a taxonomy based on observed behaviour. Table 2.3 provides a breakdown of the main classes of behaviour suggested in the proposed taxonomy. Further sub-classing (for example, a label specifying the mechanism of persistence employed) allows labelling of specific behavioural attributes in turn allowing the taxonomy to be used and expanded for categorisation over larger sample groups. The suggested naming scheme provides the class *C*, (where C = {E,D,M,S} from the classes in Table 2.3), and the subset of observed behaviours B_C to form a constructed label based on observed behaviours within classes, for example $C1_{B_{C1}}C2_{B_{C2}}C3_{B_{C3}}$ to describe a given sample within the taxonomy (Grégio *et al.*, 2015).

Table 2.3: Taxonomy of behaviours, adapted from Grégio *et al.* (2015)

Class	Behaviour Examples
Evasion	Removal of evidence or registry entries, AV termination, Firewall Termination, Update termination
Disruption	Service or port scanning for known vulnerabilities, email transmission, disruptive network activity
Modification	Modifications of files, binaries, creation of mutexes, persistence, addition of new malware
Stealing	Theft or exfiltration of data, credentials or other information

While the approach of using behaviour to group is not new, as an example Rieck *et al.* (2008) used automated analysis and learning of behaviours to group and categorise and group malware into families, naming and grouping within specific, articulated classes of behaviour may provide a method of taxonomic comparison across operating systems.

2.5 macOS: operating system, security and malware

Apple's Mac macOS (previously known as OS X and Mac OS X) operating system has traditionally enjoyed less popularity in the realm of mass marketing computing. With the rise of personal computing, companies and individuals adopted Microsoft's Windows operating system as the dominant technology of the day, with the Apple's hardware and operating system experiencing a renaissance of sorts after the release of the macOS operating system in 2002. In recent years, Apple hardware has captured significant market share (StatCounter, 2017) with a majority market share in some categories: for example, a survey by Stack Overflow, a popular question and answer online community for software developers placed macOS as the single most popular desktop operating system in use by full time developers (Stack Overflow, 2016).

Together with macOS historically shipping with more secure operating system defaults, the overall impedance in popularity and market share has, in part, led to malware being more prevalent in the Windows operating system environment than macOS. As previously noted, the motivation for writing malware may range from financial or political gain to nuisance value alone, and malware authors have taken advantage of the economies of scale found in the broad enterprise and consumer use of Windows, often enabled by frequently outdated operating system versions, poor patching and updating habits of users and dearth of available vulnerabilities (Johnston *et al.*, 2016). Given this, by far the largest opportunity to gain access to victims machines has been through the development of Windows specific malware, leading to a common perception that macOS may be largely immune to the malware that has plagued users of the Windows operating system. This has, as may be expected, led to not only well developed tooling, but a significant body of research on tools, techniques and taxonomies of Windows based malware (Egele *et al.*, 2012; Sikorski and Honig, 2012; Grégio *et al.*, 2015), with limited attention being paid to macOS malware.

Apple's operating system, however, is far from immune. As previously noted, one of the first examples of a computer virus was the *Elk Cloner*, targeting Apple IIs (Spafford *et al.*, 1991). With the growth in popularity of Apple hardware, it follows logically that more focus will be placed by interested parties on malware able to infect and persist in the macOS environment. Anti-virus vendors have previously noted significant increase in macOS malware found in the wild. Kaspersky (2014) discusses the relative sophistication macOS malware, and Carbon Black (2015) notes the significant increase in macOS malware seen in 2015, with a focus on detection and persistence methods. Using the annual datasets provided by the National Vulnerability Database[10] (NVD), the security of macOS relative to its Windows desktop operating system counterparts can be compared. Figure 2.2 shows a breakdown by year of published vulnerabilities (Common Vulnerability and Exposures, or CVEs) for both macOS and Windows desktop operating systems, including Windows XP, Windows 2000, Windows 7, Windows Vista, Windows 8 and Windows 10. The years

[10]`https://nvd.nist.gov`

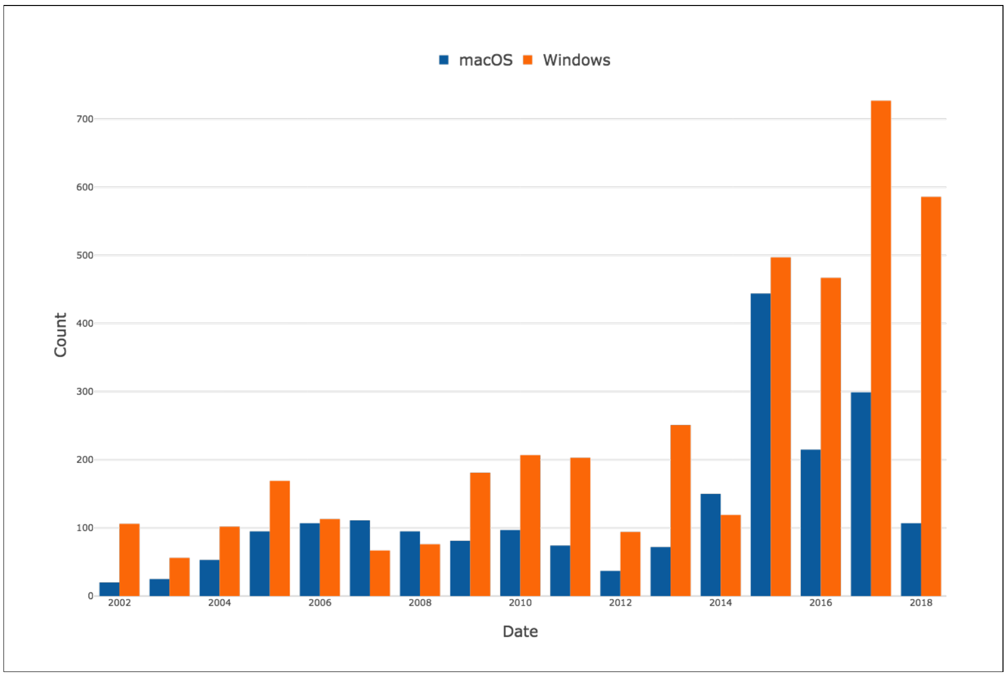


Figure 2.2: CVEs by Desktop Operating System

2007, 2008 and 2014 show more CVEs logged for the macOS operating systems of the time: as context, 2005 saw the first move of macOS from the PowerPC architecture to Intel x86 based processors, and Windows 7 was released in 2009.

While the number of CVEs may provide an indication of relative operating system security, the severity of the vulnerabilities identified should also be considered. Figure 2.3 provides a breakdown of the identified CVEs, including an indication of the NVD assigned severity rating (high, medium and low) for each of the operating system groups. While Windows based operating systems show a far higher count of CVEs in recent years, a greater proportion are considered 'low' severity vulnerabilities. macOS vulnerabilities can be seen to be predominantly 'medium' and 'high' level, with the most significant proportion of 'high' level vulnerabilities between 2015 and 2017. Overall, while an argument can be made that the macOS operating system may, at a given point in time, have been more secure by default, the assertion that the operating system is consistently more secure overall is not necessarily true.

From a technical perspective, related work exists primarily in practice. Yonts (2009) provides a high level review of tools and methods to analyse macOS malware. Baumgarten (2013) specficially analysed tools available to examine the Mach-O file format, macOS's executable file format equivalent to the PE file format used in Microsoft operating systems. Hsieh *et al.* (2016) demonstrated automatic classification methods using a sample of

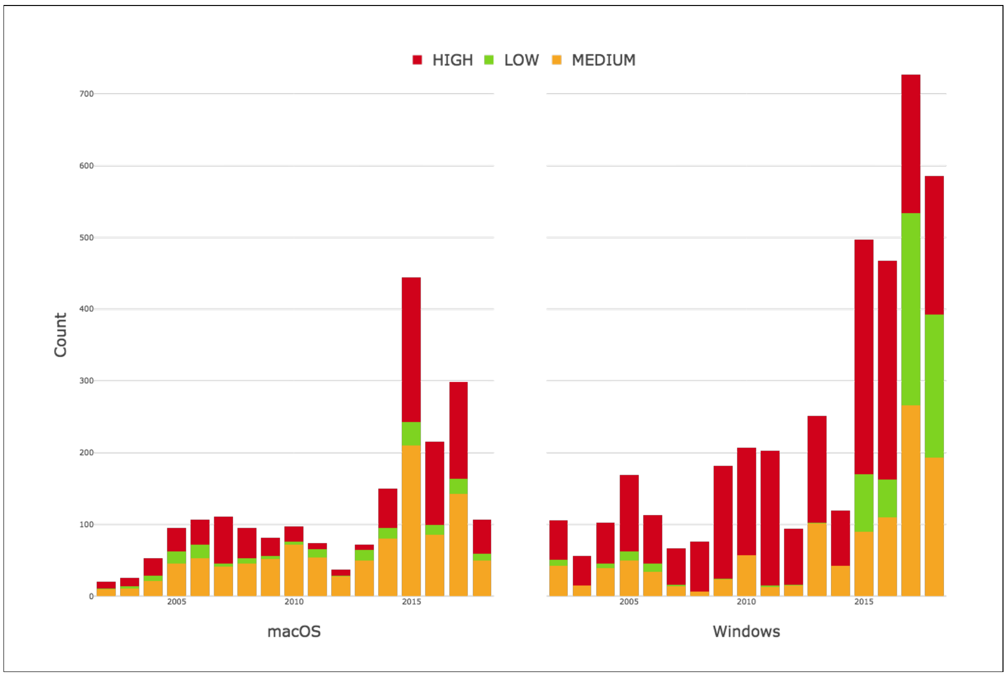


Figure 2.3: CVE by operating system and severity

available macOS malware, Van Mieghem (2016) used macOS system calls to detect malware and Pajouh *et al.* (2017) used machine learning to classify macOS malware based on dynamic analysis of library calls. macOS provides ample opportunities for persistence and infection vectors, from the *DMG* images used to distribute software (Levin, 2018) to numerous methods of persistence demonstrated by Wardle (2014) and Wardle (2017a).

Overall, the macOS operating system has been the focus of less malware infection, research and attention for reasons more likely related to overall market share than technical security prowess. However, as the operating system has been established in popular use and the constantly changing landscape provides different incentives for malware authors, attention may well turn to macOS. From a research perspective, the relative lack of scholarly attention on macOS malware provides opportunity to apply prior work on other operating systems to better understand the direction that macOS malware is likely to follow.

2.6 Summary

In this chapter, background information and work related to malware research was reviewed. The analysis, detection and evolution of malware was discussed. Important factors to consider when observing malware in the wild were covered, including common terminology in use, methods of infection and persistence and the prevalence and efficacy of code signing. Finally, specific attention was given to macOS security. In the chapter that follows, an overview of the methodology used in this research will be discussed.

Chapter 3

Methodology

I may not have gone where I intended to go,
but I think I have ended up where I needed to be.

Douglas Adams, The Long Dark Tea-Time of the Soul

3.1 Introduction

As noted by Soto-Valero and González (2018), there is a paucity of work that considers the diversity of malware on specific platforms and specifically longitudinal studies that consider empirically observed infections. The goal of this research is to contribute to this body of knowledge by addressing a gap in the study of real world infection datasets of macOS malware. By understanding diversity in an observed population of macOS endpoints, this work also creates an opportunity to further understand the evolution of malware on the platform and understand the potential future direction it may develop. Malin *et al.* (2008) suggest three broad analysis techniques for malware: temporal, relational and functional analysis.

Temporal analysis focuses on activity over a period of time, relational the interaction between different parts of a malicious activity and functional how the activity manifests. Building on this, Rossow *et al.* (2012) note discrete avenues of malware research: studies of the detection of malware, studies of longitudinal behaviour and validation of prior results. Additionally, the authors provide a number of guidelines for prudent malware research which is "correct, realistic, transparent, and do(es) not harm others" (p. 65). Further guidance suggests developing a formal methodological approach and providing clear details of the approach, collection, experimental setup and treatment of gathered data to ensure transparency.

Following this advice, this chapter will focus on these aspects: the sections that follow provide details of the overall approach taken, the setup of the experiment itself, the tools employed, the data import and cleaning process and the enrichment and categorisation carried out. Finally, validation of the data is discussed.

3.2 Approach

In keeping with the suggestions of Rossow *et al.* (2012), this research benefits from data obtained from a 'real world' operational context, and provides details of the experimental setup. As the desire is to explore and identify potential hypotheses for further research, an exploratory data analysis (EDA) approach originally developed by Tukey (1977) and expanded by Velleman and Hoaglin (1981) and others is taken. This approach has been used for exploration and analysis of security related structured data in other research (Thonnard and Dacier, 2008; Barrera *et al.*, 2010), observed malware datasets (Vermeulen, 2018) and in information security data analysis practice (Collins, 2017). The approach is flexible, with a focus on graphical representation of data to assist with identification of patterns in addition to descriptive statistics (Brillinger *et al.*, 2002).

3.3 Experimental Setup

Real world infection data from a population of endpoints, and specifically macOS endpoints, can be a challenging body of data to obtain. Potential sources of observed infections may include large corporate networks or anti-malware software vendors, however due to the commercial sensitivity of the data these are not frequently available for general research. For this research, a dataset of reported endpoint events from a population of macOS clients, along with the requisite permission to analyse the data, was obtained from an anti-malware software vendor which was then imported and cleaned (Section 3.5), aggregated (Section 3.6), enriched (Section 3.7) and categorised (Section 3.8) before being analysed in Chapter 4. This primary dataset covers suspicious events reported by an installed anti-malware client over an eleven month period from August 2017 through to June 2018. The anti-malware client on the endpoint monitors filesystem and network activity, generating events when potentially suspicious activity or a known malware signature is detected. The events are then recorded, with each record containing a reference to the file generating the event in the form of a computed cryptographic hash of the file using the Secure Hash Algorithm 1 (SHA-1) as well as potential categorisation of the file by the vendor and other related information. We refer to these as 'events', as not all are malicious, and not all constitute an infection. This is in line with the use of 'encounters' in similar work from a corporate malware event dataset by Yen *et al.* (2014).

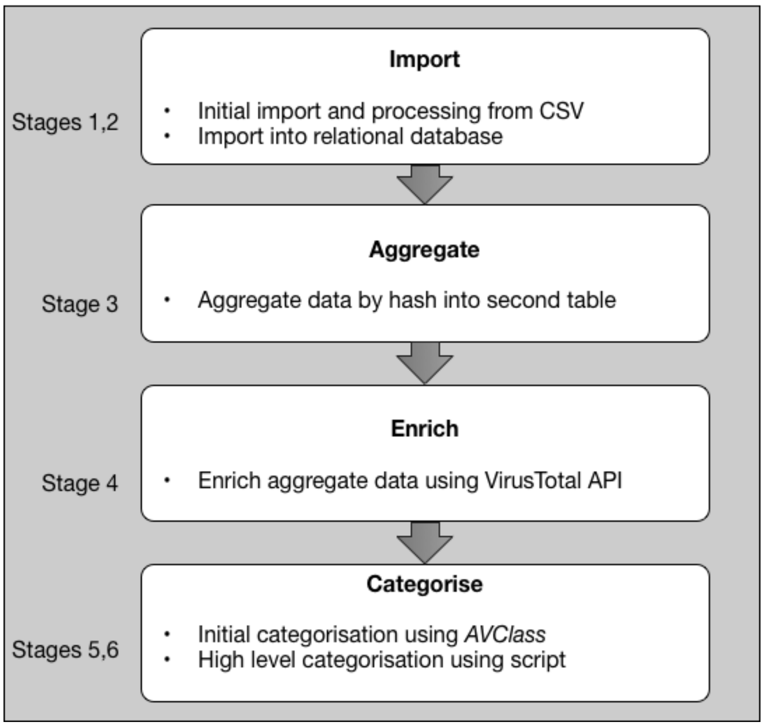

Stages 1,2
Import
• Initial import and processing from CSV
• Import into relational database
Stage 3
Aggregate
• Aggregate data by hash into second table
Stage 4
Enrich
• Enrich aggregate data using VirusTotal API
Stages 5,6
Categorise
• Initial categorisation using *AVClass*
• High level categorisation using script

Figure 3.1: Data import, enrich and categorisation process

General datasets for malware analysis are available from a number of sources, and frequently form the basis of research into malware detection, behaviour and other analysis. In common use, and discussed briefly in Section 2.4.1, is VirusTotal, an online service dedicated to malware analysis intended to function as an industry wide tool for collaboration in malware and related research. This research made use of VirusTotal under an academic access license to enrich data from the primary dataset, and assist with categorisation and aspects of temporal analysis of observed infection data, such as dates that a sample was first observed. To do this, the unique file hashes from the primary dataset were matched against the VirusTotal online service, and additional metadata recorded against each unique hash. This allowed the analysis to take into account information that may have been potentially unknown at the initial time of analysis by a single anti-malware tool, as well as any relevant categorisations by different anti-malware vendors and clustering information by VirusTotal. Exploratory data analysis was then carried out against the resulting combined and enriched dataset, which is detailed in Chapter 4. The sections that follow provide details of the steps taken, summarised in Figure 3.1, with additional details in Table 3.1.

Table 3.1: Overview of the steps taken to setup the data for analysis

Stage		**Number of Records**
1	Recieve initial dataset	1,991,861
2	Import initial dataset	1,990,182
3	Extraction of unique hashes	8,112
4	Unique hashes enriched by VirusTotal	5,676
5	Unique hashes categorised by AVClass	3,172
6	Unique hashes assigned high level category	3,254

3.4 Tooling and Environment

All work was carried out using a standard macOS workstation configured for regular software development. The workstation had limited customisation outside of the shell environment. Initial imports were carried out using iPython and the Jupyter Notebook to allow for rapid manual review of dataset samples. PostgreSQL version 9 was used to store, query and perform high level reviews of data through manual querying where required. Python version 3 was used to run scripts required for enrichment and categorisation, making use of a number of libraries including the *psycopg2* and *VirusTotalApis* packages for connecting for PostgreSQL and VirusTotal respectively and *Numpy* and *FuzzyWuzzy* packages for mathematical functions and fuzzy matching used to categorize malware types. The *Pandas* Python library was also used for initial data import.

For reporting and graphs, R[1] was used along with the *exploratory.io*[2] visualisation tool. The tool is a commercial wrapper around R that allows direct connections to a number

[1]`https://www.r-project.org`
[2]`https://exploratory.io`

of data sources including databases, and visualisation and exploration of the results. An academic license was provided by the publishers for research purposes.

3.5 Data processing: Import

The original dataset comprised of eleven separate files, with each file representing a months worth of data ranging from August 2017 to June 2018, collected from 12,601 unique endpoints. No specific geographic information was provided with the dataset, however the deployment of the software client is worldwide. The dataset was provided in comma separated value (CSV) format with 68 fields. Many of these supplied fields were vendor specific and not relevant to the research, and thus were not included for further processing and analysis. Examples of these fields include internal identifiers, client version numbers and any references to the vendor's commercial clients, for example client names that could be inferred from full file paths. The columns listed in Table 3.2 were selected for import.

Table 3.2: Vendor Fields for Import

Field Name	**Description**
ID	Unique vendor row ID
Date	Timestamp
SHA	SHA-1 Message Digest (hash)
Status	Client determined file status (suspicious, active, blocked, resolved)
Filename	Payload filename
UUID	Unique Client Identifier
Flagged As	Vendor Family Flagging

The vendor supplied data consisted of a file for each months worth of records, with files containing between 15,000 and 500,000 rows each over a number of unique endpoints, the exact details of which are shown in Table 3.3. To accommodate the format and size of the dataset, the Python library *Pandas* was used to read and extract the relevant fields. This allowed for both high level validation and initial cleaning of the data, as well as coalescing the data into a single file for import. Listing 1 shows the commands used in an iPython environment to to select and import the data from multiple files, after which the complete dataset of 1,990,183 records could be saved to a single file for further processing, with 1,678 rows discarded in the process as malformed or incomplete.

Table 3.3: Original vendor dataset size

Export Month	**08/17**	**09/17**	**10/17**	**11/17**	**12/17**	**01/18**	**02/18**	**03/18**	**04/18**	**05/18**	**06/18**
Events	15330	58827	23093	129170	347177	54452	37019	49618	209363	527443	540369
Unique Endpoints	715	763	1167	1477	1952	1517	593	926	1331	3067	2522

Using the coalesced data file, the data was then imported in a PostgreSQL database for further analysis. The use of a relational database allowed for more efficient use of the VirusTotal API in the steps that followed, as results of API requests used to enrich the data could be persisted to the database and completed in smaller batches to stay within the API request limits imposed by VirusTotal. From the originally supplied data, a total of 1,990,182 records were successfully imported with a single further record discarded as incomplete. With the full dataset loaded, more detailed processing could take place including additional levels of cleaning, enrichment of the dataset with additional data and identification of invalid data that could introduce bias or inaccuracy, discussed in the section that follows.

```python
df = pd.concat((pd.read_csv(f, header=None, usecols=[0,1,7,9,17,35,50],
↪  dtype={50: object}) for f in all_files))
df = df.rename(columns={0: 'id', 1: 'date', 7: 'sha', 9: 'status', 17:
↪  'filename', 35: 'uuid', 50: 'flagged_as'})
```

Listing 1: Import of data

3.6 Data Processing: Aggregation

As discussed previously, file SHA-1 hashes are the primary means to uniquely identify potential malware. Hashes are repeated across the primary dataset as a single client may generate multiple events for the same file, or a different client generates events based on the same file. While the primary dataset could be augmented with data directly, this would lead to unnecessary duplication. To avoid this and maintain a more dynamic structure for analysis, a second table was extracted from the imported dataset with aggregate data based on the uniquely observed hashes.

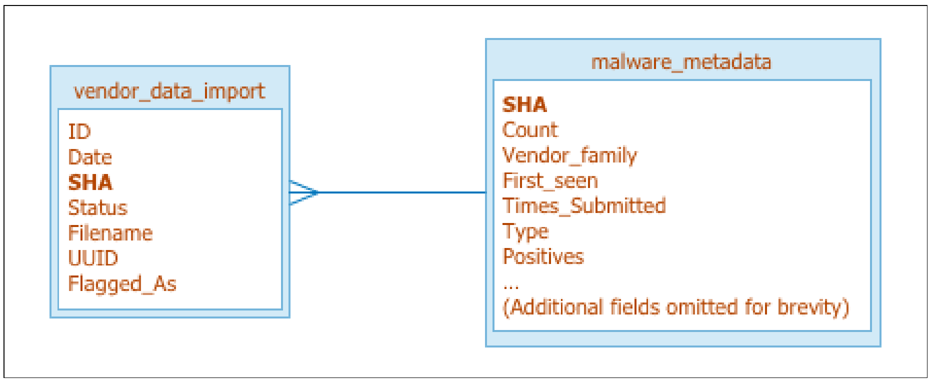


Figure 3.2: Aggregate table entity relationship diagram

For each unique hash identified in the imported vendor data, a record was created including the number of times the hash was observed, and the initial categorisation data by the anti-malware software vendor, stored in the primary data table as *flagged_as* and in the aggregate table as *vendor_family*. The SQL query used to populate the aggregate table (*malware_metadata*) can be seen in Listing 2, and the destination table structure in Table 3.4. The relationship between the table of imported vendor records and the aggregate table can be seen in Figure 3.2. On completion of the aggregation function, 8,112 records had been written to the metadata table.

```
INSERT INTO malware_metadata
SELECT
  distinct(sha) as sha,
  count(*),
  flagged_as as vendor_family
FROM vendor_data_import
GROUP BY sha, vendor_family
```

Listing 2: Aggregation of observed hashes

3.7 Data Processing: Enrichment

With the aggregate data in place, VirusTotal was then used to populate the relevant fields, seen in Table 3.4. For each record within the aggregate table, the VirusTotal API was queried, and the returned result parsed and stored if matched against a known sample. As the primary dataset contained events that may have ultimately been determined to be non-malicious, not all hashes were expected to return a result, but each was checked. By

Table 3.4: Metadata table fields

Field	**Description**	**Source**
sha	File SHA-1 hash	Primary Dataset
count	Number of events in primary dataset for this hash	Primary Dataset
vendor_family	Family name assigned by primary dataset vendor	Primary Dataset
first_seen	Date file was first seen on VirusTotal	VirusTotal
times_submitted	Number of times file has been submitted	VirusTotal
type	File type (executable or document format)	VirusTotal
positives	Number of AV engines reporting file as malware	VirusTotal
unique_sources	Number of unique sources that have sent the file in the past	VirusTotal
ssdeep	SSDeep cluster string	VirusTotal
last_seen	Most recent submission of the file	VirusTotal
processing_result	Result of enrichment process (processed or skipped)	Research
other_family	Family name(s) assigned by other AV engines	VirusTotal
classified_as	High level classification (adware, RAT, miner or ransomware)	Research
classification_scores	Recorded matching scores from the classification process	Research
avclass	Family classification	Sebastián *et al.* (2016)

the same token, the primary dataset may not have considered a particular file as malicious and thus have no category or malware information, however it may have determined at a later date to be malicious and should therefore be tagged as such. The original VirusTotal Javascript Object Notation (JSON) format report was also stored in a separate table using the native JSONB data type in order to assist with data categorisation and validation at a later stage. This table, like the aggregation table, could be linked using the file hash.

The enrichment process was performed in batches of 1000 records to assist with monitoring the completion of each batch. To reduce the likelihood of duplication in case of network error or API failure, each record was marked as processed or skipped once complete depending on the status returned by VirusTotal and excluded from future processing. The Python script used to enrich the data can be seen in the research code repository as 'Enriching the dataset with VirusTotal'[3]. On completion of the enrichment process, 5,676 or 70% of records were identified by VirusTotal, and 2,436 or 30% of records did not have a matching record on VirusTotal. It should be noted that identification by VirusTotal does not necessarily imply that a file is malicious, simply that the file hash had been submitted to VirusTotal and been processed at a given point in time. In these cases, available metadata was populated (for example, file type) however no positive identifications (the *positives* field) or family data seen in Table 3.4 were recorded.

3.8 Data Processing: Categorisation

As discussed in Chapter 2, earlier malware definitions made use of infection vectors (for example: *viruses*, *worms*, and *trojans*) to assist with categorisation. Contemporary approaches have evolved to consider both the infection vector and outcome (for example, *ransomware*, *spyware* or *keyloggers*). Ultimately an exploration of malware data could benefit from higher level grouping and categorisation in order to understand the overall diversity of threats impacting the platform. In the previous steps the granular activity logs were imported, and the unique hashes extracted and enriched with data from VirusTotal to aid further analysis. As a final step in the enrichment process, each of the hashes identified as malicious are grouped into family categorisations with *AVClass*, and then further grouped into four discrete high level categories to assist with reporting and analysis.

Categorisation took place in two stages. In the first stage, prior work by Sebastián *et al.* (2016) was used to perform an automated classification of the aggregated dataset. The

[3]`https://thesis.yaxs.net/posts/enriching_the_dataset`

result of their work, the *AVClass* malware classification tool[4], was a suitable for a number of reasons. Firstly, the tool can be considered sufficiently accurate, having been evaluated in their research against multiple malware datasets with ground truth. Secondly, the tool removed the need for manual analysis of a large dataset. Finally, the tool was able to make use of VirusTotal reports in a manner that is vendor neutral, allowing this research to make use of it with no additional tooling or data synthesis. To carry out the categorisation, the VirusTotal JSON file report was extracted and made available to AVClass; the report and vendor classifications were then evaluated by the tool and the corresponding field updated on the aggregate table (*avclass*, as seen in Table 3.4) with a resulting 3,172 records successfully classified.

For the second stage of categorisation, both the existing research and contemporary industry reports previously cited and a manual review of the *AVClass* identified families were considered in order to arrive at high level categories that could be used for later analysis. Specifically noted groups found were adware and PUPs, cryptocoin miners, ransomware and backdoors, trojans, and remote access tools (RATs) (Symantec, 2018a; Kaspersky, 2018; Reed, 2018b; Wardle, 2018b).

The first category (Category A) includes adware and potentially unwanted programs or applications (PUPs or PUAs). Urban *et al.* (2018) bundles Adware and PUPs together, noting significant privacy and security implications of both. Both involve deceptive installation techniques, with a defining characteristic being inadequate user notification of the true intent of the software or additional software that may be installed, usually involving non-consensual tracking of user behaviour and installation of behaviour changing internet browser behaviour. PUPs may also be bundled with legitimate software through pay per install (PPI) services (*ibid.*). Example of Category A malware include *MacKeeper* (Reed, 2016b) and *Bundlore* (Malwarebytes, 2015).

The second category (Category B) includes remote access tojans (RATs), backdoors and command and control (CnC) clients used by botnets. The defining characteristic of this category is that the software is installing unknowingly by the user, usually by taking advantage of an exploit triggered by running a file that may have been downloaded, emailed or included in compromised software installers. The motivation for creating malware included in this category may not be immediately apparent and could range from theft of data or credentials, to illegal spying activities. Where Category A malware may masquerade as legitimate software, category B software does not, and is more aligned to traditional malware methods of infection and persistence. Examples of Category B

[4]`https://github.com/malicialab/avclass`

malware include *Flashback* (Bureau, 2012), *MacControl* (F-Secure, 2012a) and *GetShell* (F-Secure, 2012b). Attackers my use this category of malware for activities ranging from keylogging and spying on users, to remote control of their machines to exfiltrate data or participate in botnet activities.

The third category (Category C) includes ransomware or crypto locking malware. These may use many of the infection vectors of Category B, but the defining characteristic of this category of malware is the overt commercial imperative and highly disruptive nature. This category carries out an activity for immediate financial gain: locking access to files in the hope of extorting a payment to decrypt or regain access to files. Ransomware has been studied extensively (Gazet, 2010; Kharraz *et al.*, 2015) and the results of ransomware have caused significant commercial losses in recent years (Mansfield-Devine, 2017). An example of this category within the primary dataset would be *KeRanger* (Xiao and Chen, 2016).

Finally the forth category (Category D) includes cryptocurrency miners. Sharing both likely infection vectors and motivation (an immediate commercial imperative) with Category C malware, it is distinguishable by the lack of visible disruption to the user environment outside changes in system performance, and limited visible evidence of operation. Relatively few examples of this malware were observed in the dataset, one being *Coinminer* (Latif, 2018).

Table 3.5: Keywords used in fuzzy matching family names

Category	**Type**	**Keywords**
Category A	Adware and PUPs	pup; pua; adware; cleaner; keeper; opinion
Category B	RATs and backdoors	backdoor; hack; shell
Category C	Ransomware	ransom; crypt
Category D	Cryptocurrency Miners	miner; coin

The final classification was itself a two-stage process involving scripted, automatic classification based on the vendor assigned family names, and a manual sampling and review to validate that the correct family had been assigned. Based on insights from the prior categorisation, a collection of keywords was assembled that represented each of the categories.

These keywords are shown in Table 3.5. The *FuzzyWuzzy*[5] Python library was used to compare each of the keywords against the collection of assigned family names from both the primary data (i.e., the vendor assigned family name) as well as the VirusTotal dataset. The library is commonly used for string comparison in a number of different languages, utilising an implementation of a Levenshtein Distance (Levenshtein, 1966) to score and match compared strings. Specifically, the library allows for use of token sets which are insensitive to duplication: a common feature of combined family name strings across anti-malware software vendors.

Thus, in the following example a collection of assigned family names for the SHA-1 file hash starting *6efdfd* are compared to the keyword library (Table 3.5) and scored using a token set comparison. This yields a more realistic score as the numerous duplicate terms (for example: variant, application or generic) are reduced to single entries, which are then sorted and compared.

Vendor labels for hash 6efdfd21e94dc04d6f3d60ed7337cd90b67004d7 from VirusTotal:

```
MacOS.PUA.Mackeeper; Gen:Variant.Application.MAC.PazaCA.1; RDN/
   Generic.osx; Trojan.Application.MAC.PazaCA.1; OSX.MacKeeper!
   gen1; a variant of OSX/Mackeeper.A potentially unwanted;
   Suspicious_GEN.F47V0614; Osx.Malware.Agent-6490746-0; Gen:
   Variant.Application.MAC.PazaCA.1; Gen:Variant.Application.MAC.
   PazaCA.1; Gen:Variant.Application.MAC.PazaCA.1 (B); Gen:Variant
   .Application.MAC; RDN/Generic.osx; Generic PUA JF (PUA);
   malware (ai score=95); PUA:Win32/Presenoker; malicious (high
   confidence); Gen:Variant.Application.MAC.PazaCA.1 (2x); PUA.OSX
   .MacKeeper; Adware/Generic_PUA_JF
```

The result for the example above shows **Category A** (adware and PUPs) as the overall winner, which is correct for the identified malware (*MacKeeper*):

```
{"Category B": 20.75,
 "Category D": 22.3,
 "Category A": 28.0,
 "Category C": 16.85}
```

For the purposes of data processing and record keeping, both the winning category and scoring detail (in the same format as the example above) were stored against each of the hashes in the rest of the data. The Python source code for the script used to classify the

[5] `https://github.com/seatgeek/fuzzywuzzy`

relevant records is viewable in the research code repository as 'Categorising the dataset into four main categories'[6]. Examples of the input and categorisation output can be found in the same repository as 'Examples of categorised metdata rows'[7].

3.9 Data Validation

Validation was carried out at each step of the process. An initial review of data included a simple timeline analysis to validate that no unexplained gaps were present in the data, and an endpoint analysis to validate that the dataset included activity from a reasonable number of unique endpoints. Table 3.3 provides a breakdown of the data by both month of import and unique reporting clients, which where grouped by the endpoint Universally Unique Identifier (UUID) included in the initial import.

Once coalesced, the data was scanned for field completeness of key fields and any fields without key fields (in this case, SHA-1 hash, date and endpoint UUID) excluded from import into the relational database. The previously noted Table 3.1 indicates a removal of 1,678 incomplete records from 1,991,861 initially received, an acceptable total of 0,084% of the primary dataset. As further aggregation was completed within a relational database and by script, the validation of the enrichment and categorisation activities was considered as part of the running of the scripts themselves, with the total rows effected also noted in the previously listed Table 3.1.

One exception to this was noted during a check of observed events by endpoint, which identified that certain endpoints had an unusually large distribution of malware events generated. On closer examination it was determined that these endpoints were used in test environments either by the vendor, or independent researchers using the vendors products. In many cases the endpoint hostnames could be determined from the full filepath of the file identified, providing this researcher with an effective way of identifying and tagging endpoints likely to have been used for testing, thus avoiding later skewing of results. A similar observation was noted by Yen *et al.* (2014), who also excluded research activities from their results.

Examples of hostnames identified as likely testers included 'macsdontgetmalware', 'infosec', 'virustotal' and hostnames that included the name of the anti-malware vendor company itself. To ensure completeness, records were not removed but the additional

[6]`https://thesis.yaxs.net/posts/classifying_the_dataset/`
[7]`https://thesis.yaxs.net/posts/categorisation_example/`

field *known_tester* added to the primary vendor dataset, and flagged as true for any record generated by a likely test endpoint. A total of 889 records of 1,990,182 were flagged this way.

Finally, a manual review of the data was carried out by the researcher. Consistency was checked between assigned labels and categories, and in a small number of cases, incorrectly classified non-adware malware was identified and corrected. Rows manually updated in this way were also updated with a *manually_updated* flag. A total of eleven rows were updated this way.

3.10 Summary

In this chapter the methodology and experimental setup were discussed. The vendor dataset was imported, cleaned and processed, enriched with VirusTotal data and finally categorised in two stages: initially using AVClass, followed by a script developed to assign a high level category in line with manual review and categories from current industry reporting. In the chapter that follows the data will be explored and analysed. Analysis will focus on temporal factors, observed malware families and categories, diversity of the observations, code signing and consensus between the various anti-malware software vendors that provide input into VirusTotal for the malware samples observed.

Chapter 4

Analysis

The trouble with having an open mind, of course, is that people will insist on coming along and trying to put things in it.

Terry Pratchett, Diggers

4.1 Introduction

In Chapter 3, the dataset was cleaned, aggregated and enriched to allow for the exploratory data analysis which follows. Exploratory Data Analysis (EDA) is summarised by Collins (2017) as an analysis of activity data with no preconceived assumptions about either the data, or the behaviour it represents. The ultimate goal of EDA is to move towards some form of model, be it a formal representation of the data, or an established baseline that can add value to future research. Along with exploration and visualisation of the data, domain knowledge and contextual information should be used to interpret the data. "The analysts judgement and and circumstances surrounding the data also play important roles" (Velleman and Hoaglin, 1981, p. 16), so application of technical knowledge from related areas should allow non-expert readers to quickly understand implications and possible findings.

The direction of exploration for this research is drawn from existing work undertaking analysis of platform specific malware datasets for non-macOS platforms, specifically Microsoft Windows (Yen *et al.*, 2014), and the Android mobile platform (Soto-Valero and

González, 2018; Hurier *et al.*, 2016). Additionally, industry reports and studies from practice are considered, such as Symantec (2018a), Proofpoint (2018) and the macOS specific works by Wardle (2018b) and Reed (2018b). From these works, a number of exploratory themes emerge:

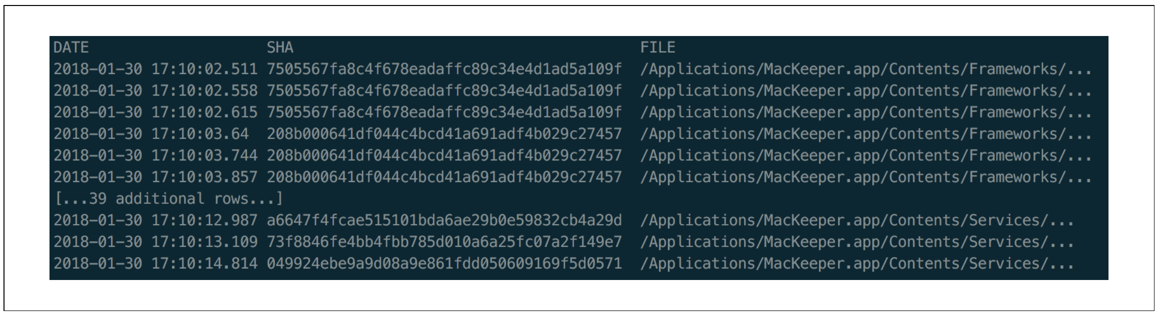
1. Analysis of temporal factors (Section 4.2)
2. Observed malware families and types (Section 4.3)
3. Diversity within and clustering of the observed malware population (Section 4.4)
4. Infection and persistence mechanisms and operating system controls in place to ensure integrity and authenticity of software, such as code signing (Section 4.5)
5. Consensus between multiple anti-malware software vendors (Section 4.6)

The sections that follow provide a exploration of the dataset within in these common themes. The chapter concludes with a summary in Section 4.7.

4.2 Time Series Analysis

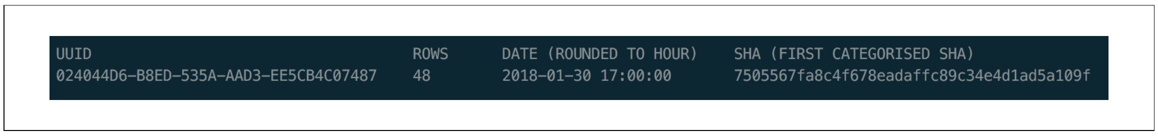
The nature the anti-malware engine used is such that multiple events were reported against the same infection, and where potentially malicious software with multiple libraries were detected, as seen in adware, that a single infection may be reported as multiple files that make up the application are detected as malicious. Thus, it was required that data first be grouped both by the time an infection was first observed, as well as the high level AVClass or category (the assignment of which was discussed in Section 3.7) in order to derive meaningful summary statistics.

An example of this process is can be seen in Figure 4.1, showing a single observation on a single client, spanning 48 rows over a 12 second period. Many of these rows are libraries and files that may be related to the same observed malware incident. In order to accurately analyse this, this is reduced to a single row selecting the first instance of a categorised observation for that specific date and time, allowing multiple detected files on the same host at the same time to be considered a single observation. The end result of this example can be seen in Figure 4.2.



```
DATE                    SHA                                       FILE
2018-01-30 17:10:02.511 7505567fa8c4f678eadaffc89c34e4d1ad5a109f  /Applications/MacKeeper.app/Contents/Frameworks/...
2018-01-30 17:10:02.558 7505567fa8c4f678eadaffc89c34e4d1ad5a109f  /Applications/MacKeeper.app/Contents/Frameworks/...
2018-01-30 17:10:02.615 7505567fa8c4f678eadaffc89c34e4d1ad5a109f  /Applications/MacKeeper.app/Contents/Frameworks/...
2018-01-30 17:10:03.64  208b000641df044c4bcd41a691adf4b029c27457  /Applications/MacKeeper.app/Contents/Frameworks/...
2018-01-30 17:10:03.744 208b000641df044c4bcd41a691adf4b029c27457  /Applications/MacKeeper.app/Contents/Frameworks/...
2018-01-30 17:10:03.857 208b000641df044c4bcd41a691adf4b029c27457  /Applications/MacKeeper.app/Contents/Frameworks/...
[...39 additional rows...]
2018-01-30 17:10:12.987 a6647f4fcae515101bda6ae29b0e59832cb4a29d  /Applications/MacKeeper.app/Contents/Services/...
2018-01-30 17:10:13.109 73f8846fe4bb4fbb785d010a6a25fc07a2f149e7  /Applications/MacKeeper.app/Contents/Services/...
2018-01-30 17:10:14.814 049924ebe9a9d08a9e861fdd050609169f5d0571  /Applications/MacKeeper.app/Contents/Services/...
```

Figure 4.1: Observation of a single MacKeeper detection on a single host



```
UUID                                  ROWS     DATE (ROUNDED TO HOUR)    SHA (FIRST CATEGORISED SHA)
024044D6-B8ED-535A-AAD3-EE5CB4C07487  48       2018-01-30 17:00:00       7505567fa8c4f678eadaffc89c34e4d1ad5a109f
```

Figure 4.2: Grouped record of the Figure 4.1 detection, rounded to the closest hour

Of the 1,990,182 event records noted in Section 3.9 *less* records generated by known testers and false positives recorded as such by administrators (a total of 739,473 records), 1,250,709 categorised event records were available to be grouped into observations. With the grouping completed, 3,450 observations were recorded.

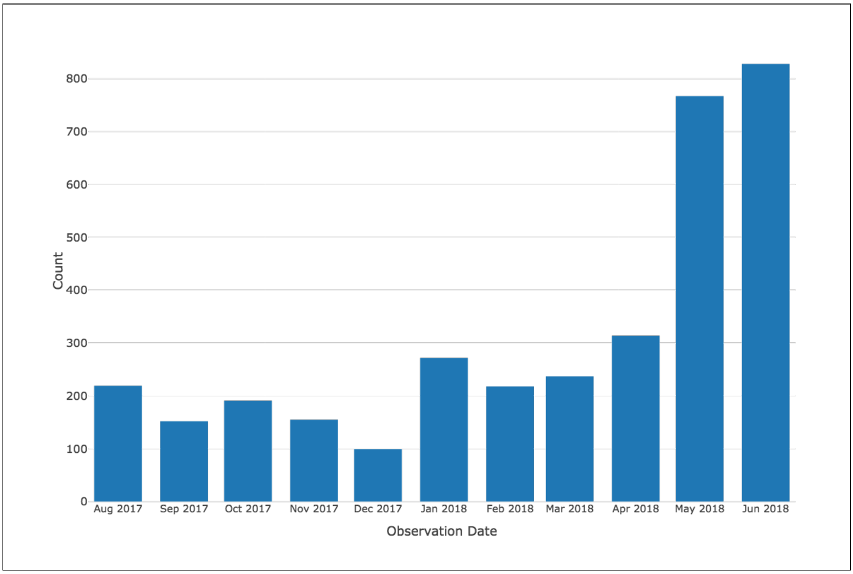


Figure 4.3: Timeline of reported malware (Categories A - D)

Figure 4.3 shows the malware events recorded over the eleven month period covered by the dataset from August 2017 to June 2018. In the initial period from August to December 2017, malware events reported remain relatively similar with the lowest recorded number of malicious software detections during December 2017. Malware events then show a strong upward trend with a peak at the end of the period (June, 2018), after a brief reduction in April 2018.

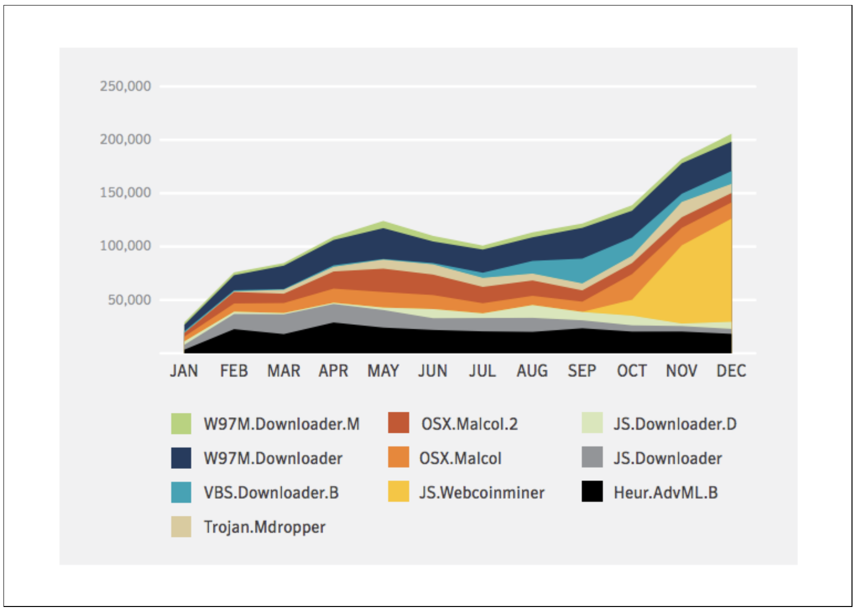


Figure 4.4: Comparison: Top 10 macOS malware blocked by month for 2018 from Symantec (2018a)

These observations are broadly in line with industry trends reported in the previous year for overall observations, with Symantec (2018a) reporting a similar monthly detection pattern climbing to a peak in April and May, and an overall drop towards the end of the calendar year. However, within the same report the number of blocked macOS specific infections shows a marked increase towards end of the calendar year in 2017 as shown in Figure 4.4, driven largely by detections of the *JS.Webcoinminer*, a multi-platform browser based cryptominer (Symantec, 2018a). When this specific threat is isolated and removed from the dataset, the detection pattern normalises in line with the overall reported detection rate in this research.

Figure 4.5 shows malware over the same timeseries, but separated by high level category, which is discussed in more detail in Section 4.3. In increase over the period under observation can be seen in both adware and PUPs and RATs, trojans and backdoors. Cryptocurrency miners have only a few observations along with sporadic observations of ransomware, leading to gaps in the timelines for these two categories and a cautious approach to drawing any conclusions.

Figure 4.6 provides of view of the dataset by day of week over the period under analysis. Saturday and Sunday show a decline in reported incidents, unsurprising for an enterprise or business focussed anti-malware tool. During the working week, malware shows a peak

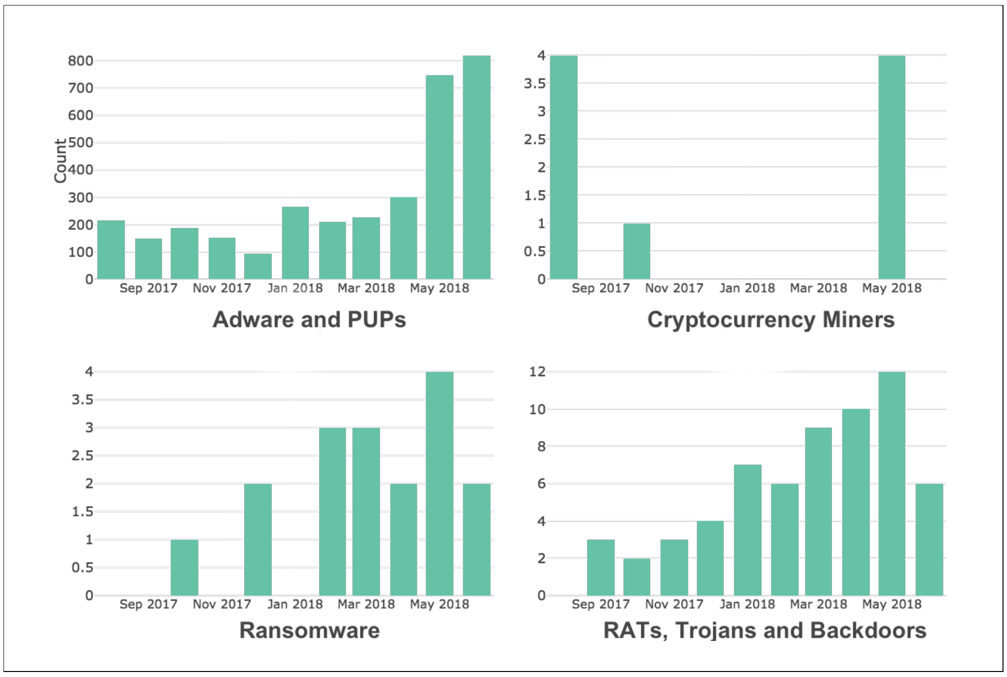


Figure 4.5: Timeline of reported malware by category

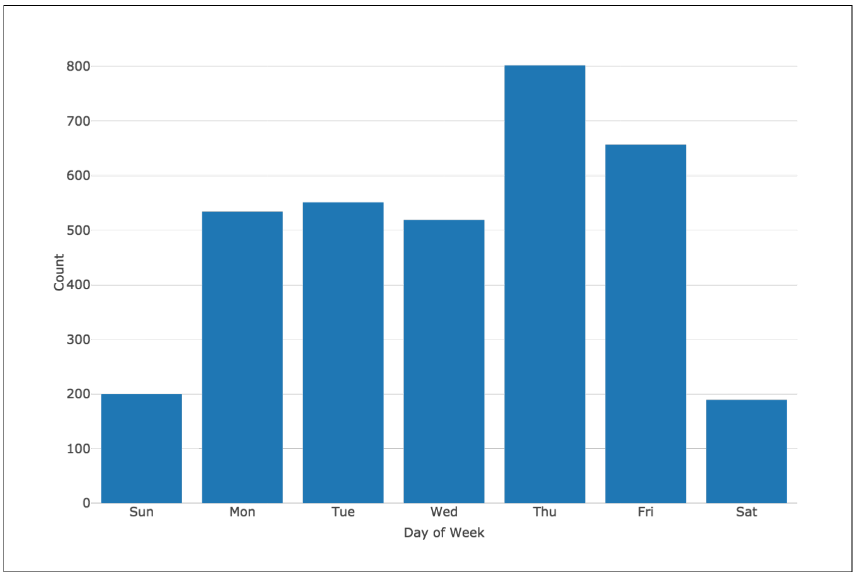


Figure 4.6: Malware observations by day of the week

on Thursdays, with weekdays reflecting a higher observation rate than weekends. A similar finding was made by eSentire (2017), who noted persistent activity between Mondays and Thursdays, trending downwards from Fridays. Figure 4.7 shows a visual breakdown of the observed malware by hour of the day. The volume remains relatively steady, climbing from midday to a peak between 17h00 and 18h00. Both the vendor source data as well as the database store the timestamps the data is derived from in Coordinated Universal Time (UTC). The peak during the afternoon period in UTC may reflect an increase in detections at the start of the business day for users in US geographies.

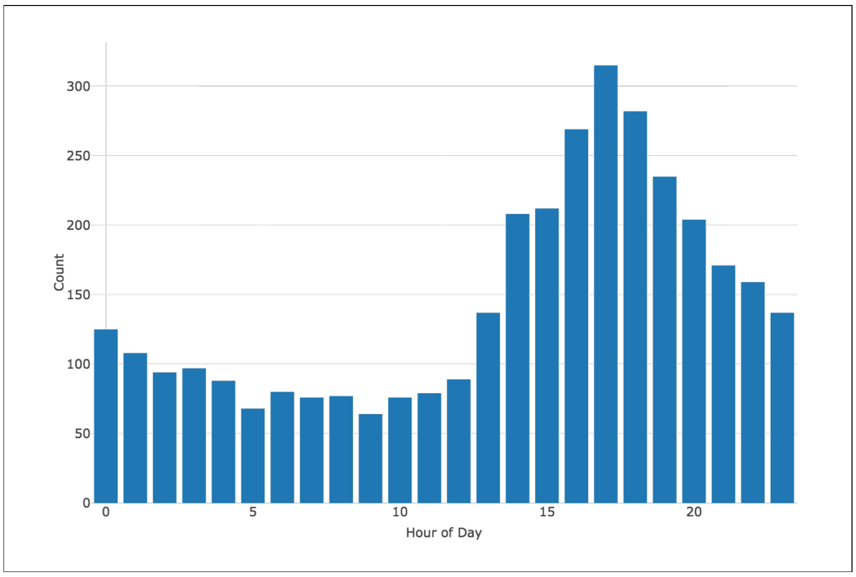


Figure 4.7: Malware observations by hour of day (UTC)

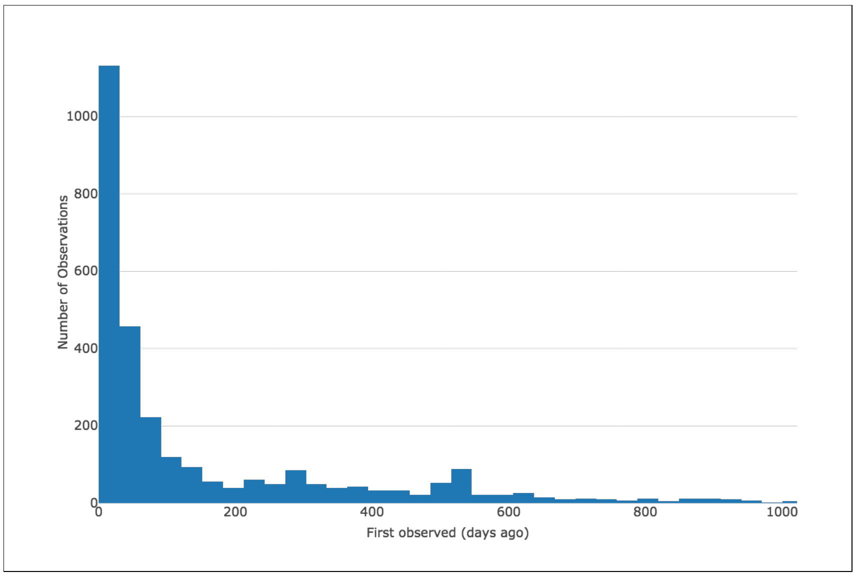


Figure 4.8: Days between recorded observation and first sight on VirusTotal

Finally, an estimation of the relative age of the observed malware is shown in Figure 4.8. The metric is measured by calculating the difference between the observation date of the malware, and the date the sample was first seen by VirusTotal from any malware vendor, analyst or public submission. 15% of malware observations were unseen at the time of submission to VirusTotal, indicating the submission was either of a previously unseen sample or, in many cases, a previously unseen SHA-1 hash of a known malware due to differing binaries or polymorphic malware (Apel *et al.*, 2009).

For the balance of the observations (85%) which *were* previously seen on VirusTotal, the values have a minimum of 0.04 days (around 1 hour), a maximum of 3,032.7 days (around 8.3 years), a median of 50.67 days, an average of 192.79 days with a standard deviation of 333.86 days. The skew in the dataset is clearly visible in Figure 4.8, with 33% of the observations occuring having been first seen within 30 days of the observation date. Malware towards the far end of the range (greater than 1,000 days) justified further exploration, with a manual review of malware with an initial observation date greater than 1,000 days prior showing that older samples were primarily non-adware malware (specifically, Category B or RATs, Trojans and Backdoors). Of note is that Category A malware (Adware and PUPs), which makes up the vast majority of observed malware in the dataset and is discussed further in the following section, only appears from 2013 onwards. The ten oldest unique initial observations are shown in Table 4.1. An EICAR test file (used to test anti-malware software signature detection) was the oldest identified from the set, having been first seen on VirusTotal with the submitted SHA1 hash in February, 2010.

4.3 Classification and Labelling

In the process described in section 3.7, records of malware activity were enriched with metadata from external sources such as VirusTotal and automated labelling work by Sebastián *et al.* (2016). This enrichment process provides valuable insight into the nature of the malware itself, and is reviewed in the analysis that follows.

Table 4.2 shows the breakdown of malware categories observed. Once grouped in the process described in section 4.2, 3450 malware observations over the period are recorded. The largest category is category A (adware and PUPs), at 97.88% of the total malware observations. This is followed by category B (RATs, backdoors and trojans) at a significantly smaller 1.68% of the total malware observations. Finally, Category C and D

Table 4.1: Top 10 oldest malware samples by VirusTotal *first seen* date

First Seen	**Observation**	**File Type**	**Category**	**AVClass**	**Days**
2010-02-04	2018-05-25	Text	Test File	eicar	3032.7
2011-04-06	2018-04-03	Mach-O	RAT	getshell	2554.7
2012-02-15	2018-05-16	Mach-O	RAT	blackhol	2282.8
2012-03-27	2018-03-07	Mach-O	RAT	fakeco	2171.7
2012-10-06	2018-03-29	Mach-O	RAT	rubilyn	2000.7
2013-02-15	2018-05-14	Mach-O	RAT	callme	1914.7
2013-04-28	2018-06-21	Mach-O	RAT	clapzok	1880.9
2013-03-03	2018-04-04	Mach-O	RAT	hellraiser	1858.8
2013-04-01	2018-01-19	Mach-O	Adware	pazaca	1754.4
2013-07-15	2018-01-01	Mach-O	Adware	genieo	1631.5

include ransomware, cryptominers and related malware and make up around 0.2% of the total each. The significant percentage of adware observed aligns with the emperical study of Android malware by Soto-Valero and González (2018), who identified that adware is the prevalent malware type of malware distributed through Android application markets.

The family of malware within the categories generated using *AVClass* is considered next. The dataset was divided into adware and non-adware malware based on the overall category, with category B, C and D malware included in the latter. As Category A was the majority, this allowed the research to consider potential differences in categorisation and diversity with malware that may arguably represent a greater practical security threat. Tables 4.3 and 4.4 show the top ten Category A and Category B, C and D classifications respectively, listed in these separate groupings.

The most frequently observed label is the *installcore* family of adware. This family of adware and its variants are frequently installed by third party software applications or by masquerading as software updates causing users to unknowingly install the unwanted program (Malwarebytes, 2018b). The second most prevalent label is *mackeeper*, an application that has been distributed for some time and is widely regarded as deceptive and misleading (Reed, 2016b). Most of the remaining labelled adware follow similar patterns of infection and persistence, with some exceptions. *Spigot* focuses on browser hijacking

Table 4.2: Breakdown of malware categories

Category	**Unique AVClasses**	**Count**	**Percentage**
Category A (adware and PUA/PUPs)	122	3363	97.48%
Category B (RATs and Backdoors)	39	71	2.06%
Category C (Ransomware)	2	7	0.20%
Category D (Cryptocurrency Miners)	4	9	0.26%
Total	**167**	**3450**	**100%**

Table 4.3: Top 10 AVCLass classifications: Category A

Rank	**AVClass**	**Observations**	**Percentage**
1	installcore	669	20%
2	mackeeper	556	17%
3	genieo	474	14%
4	spigot	347	10%
5	supportgeeks	220	7%
6	amcleaner	201	6%
7	cimpli	188	6%
8	bundlore	160	5%
9	pazaca	59	2%
10	tirrip	57	2%
		Total % of category	87%

(Arntz, 2017), and *genieo* is noteable for significantly more malicious behaviour, taking advantage of known vulnerabilities in macOS to enable privilege escalation and persistence (Reed, 2016a; Malwarebytes, 2018a). Similarly, *tirrip* (a label variation on the *pirrit* malware) uses privilege escalation, traffic hijacking and competitor software removal for commercial gain and has been noted as a particularly malicious strain of adware (Serper, 2016).

Table 4.4: Top 10 AVClass classifications: Categories B, C and D

Classification Category	**AVClass**	**Manual Classification**	**Observations**	**Percentage**
Category B	getshell	-	11	13%
Category B	macontrol	-	7	8%
Category B	rubilyn	-	5	6%
Category B	proton	-	3	3%
Category B	SINGLETON	Unknown	3	3%
Category B	SINGLETON	Trojan.WisdomEyes	3	3%
Category C	keranger	-	4	5%
Category C	macransom	-	3	3%
Category D	SINGLETON	Mac.CoinMiner	5	6%
Category D	SINGLETON	OSX.Miner	2	2%
			Total % of category	53%

Non-adware malware listed in table 4.4 highlights the significant difference in the volume of non-adware malware observed. Samples processed by *AVClass* that were considered isolated or could not be successfully allocated a family label are listed as singleton instances. The *AVClass* documentation[1] notes that singleton classes are assigned when no family name could be identified from the analysed labels. As a result, samples are tagged as singletons along with the relevant hash. The manual classification column provides additional details in these cases, following a process of manual investigation of the hash using a VirusTotal search.

[1]`https://github.com/malicialab/avclass`

The top observation, *getshell*, is a malicious Java based trojan first observed in 2013 that installs on a users device, and executes remote payloads and creates one or more administrative user accounts (Symantec, 2018b). Category B malware make up the largest number of non-adware malware observations, with limited observations of both Category C and Category D. During the collection period of the dataset, two ransomware threats were known to be prevalent (*keranger* and *patcher*), one of which (*keranger*) was observed in Table 4.4. The other ransomware sample, *macransom*, has been acknowledged as somewhat unique due to its availability as 'ransomware for hire', but viewed as largely infective (Wardle, 2017b). The relative spread of malware in this category is also noteably different to Category A, with a number of single samples making up the balance (the top ten only making up 53% of malware observed), with a number of single samples and Windows malware detected making up the balance. While the precise reasons for Windows samples being detected aren't known, it is possible that Windows malware may be detected on external storage devices shared with Windows endpoints.

4.4 Diversity and Clustering

With observations having been classified using *AVClass*, exploration of the diversity of the malware population is possible. Additionally, the additional *SSDeep* cluster identification included in the VirusTotal enrichment allow for the dataset to be clustered and visualised accordingly. These will both be considered in the sections that follow.

4.4.1 Diversity

The Shannon entropy index discussed in Section 2.4.1 can be used as a measure of diversity within a population. The work of Soto-Valero and González (2018) used a Shannon entropy index as a measure of balance within a population of malware. Direct comparison is unfortunately not possible, as in the referenced work the authors were able to measure diversity of a population by using the known population of Android malware as the total population within the diversity function. In the case of this dataset, the total known population of OS malware is not known. As a result, this research applied the diversity index simply as a measure of diversity in the population over time, using the total observed labels as the total population size for the diversity function. While limited in application, rate of change in diversity can be seen within the time period of the dataset. Similar to the previous analysis, the non-adware categories (B, C and D) are extracted in order to

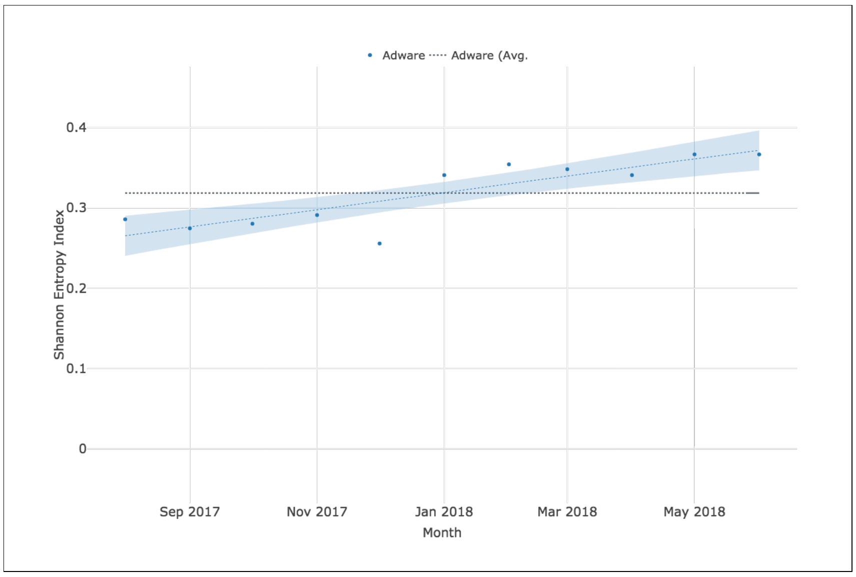


Figure 4.9: Shannon entropy index values for malware by label (all malware)

consider the diversity of the malware shown in Figure 4.3. Figure 4.9 and 4.10 provide a view of the entropy values over the period of the dataset, grouped within the relevant month.

A relatively low diversity in macOS malware is reported in practice by Wardle (2018b). When visualising the growth of observed diversity over time, a slight upward trend can be seen over the period for the full malware dataset. The much smaller non-Category A malware shows a more significant increase in diversity over the time period, however it should be highlighted that the size of a population is considerably smaller, so smaller increases in observed malware in this category have a commensurately higher impact on the result. It should be noted that Windows observations were excluded from the diversity sample calculations. While diversity is shown to be increasing in this dataset, and especially in Category B, C and D malware, a study over a longer time period may yield more conclusive results.

4.4.2 Clustering

While the process of automated labelling using *AVClass* allows the observations to be grouped by the derived family name, the enrichment of the dataset using VirusTotal provides an additional data point in the form of a computed *SSDeep*[2] Context Triggered

[2]`https://ssdeep-project.github.io/ssdeep/index.html`

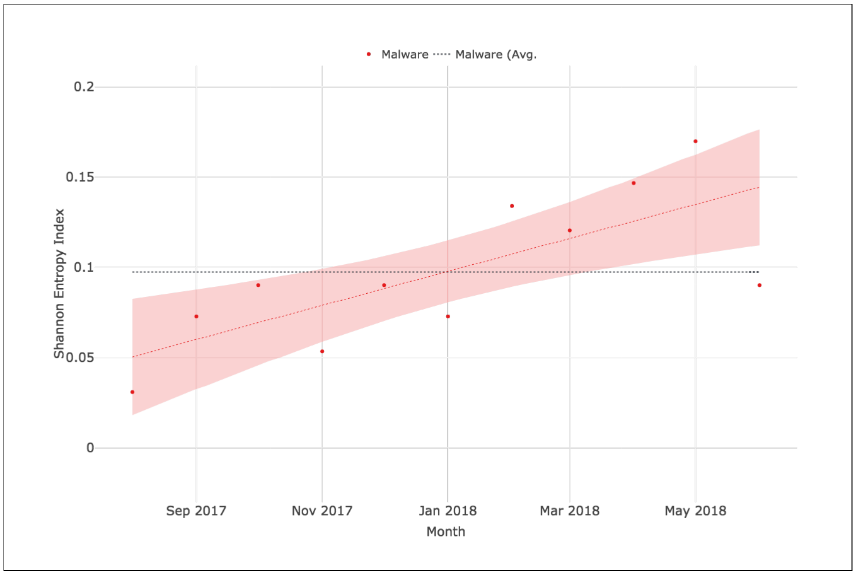


Figure 4.10: Shannon entropy index values for malware by label (non-Category A malware)

Piecewise Hash (CPTH) for 3444 of the 3450 observations (99.83%) in the dataset. An example of computed *SSDeep* hashes for two different samples are shown in Table 4.5.

Table 4.5: Example of *SSDeep* hashes for two malware samples

Sample Description	**SSDeep Hash**
MacKeeper Adware	6144:bKeOqpESonxYvfDLrzmVSgyC6CpJ3PyUwyM0H0H: +eOqpEQf2VSgy17/H
EquationDrug Trojan	384:9zpReZVUauRKt/xAEt0LzE8QXA5xGPA1N3ehU: 9vzauQry4a

The hashes consist of three parts, separated by colons, that provide the chunk or block size, and two computed values based on the chunk size. Using *SSDeep*, these hashes can be used to determine similarity and infer likely relations between samples (Kornblum, 2006). For large datasets this can be a somewhat inefficient process, as every *SSDeep* hash needs to be compared to every other other hash. Wallace (2015) provides a repeatable and scaleable method for clustering of large volumes of malware based on *SSDeep* hashes, which in turns allows for a visual representation of the identified clusters for the dataset. The result of this clustering using this method and the related SSDC utility[3] is shown in Figure 4.11. This analysis would be incomplete without mention of the limitations

[3]`https://github.com/bwall/ssdc`

of *SSDeep*: work by Pagani *et al.* (2018) demonstrates that CPTH may not always be the best choice for binary comparison. Alternatives comparisons using *tlsh* (Oliver *et al.*, 2013) and *sdhash* (Roussev, 2010) are able, in some cases, to identify similarities missed with the use of CPTH.

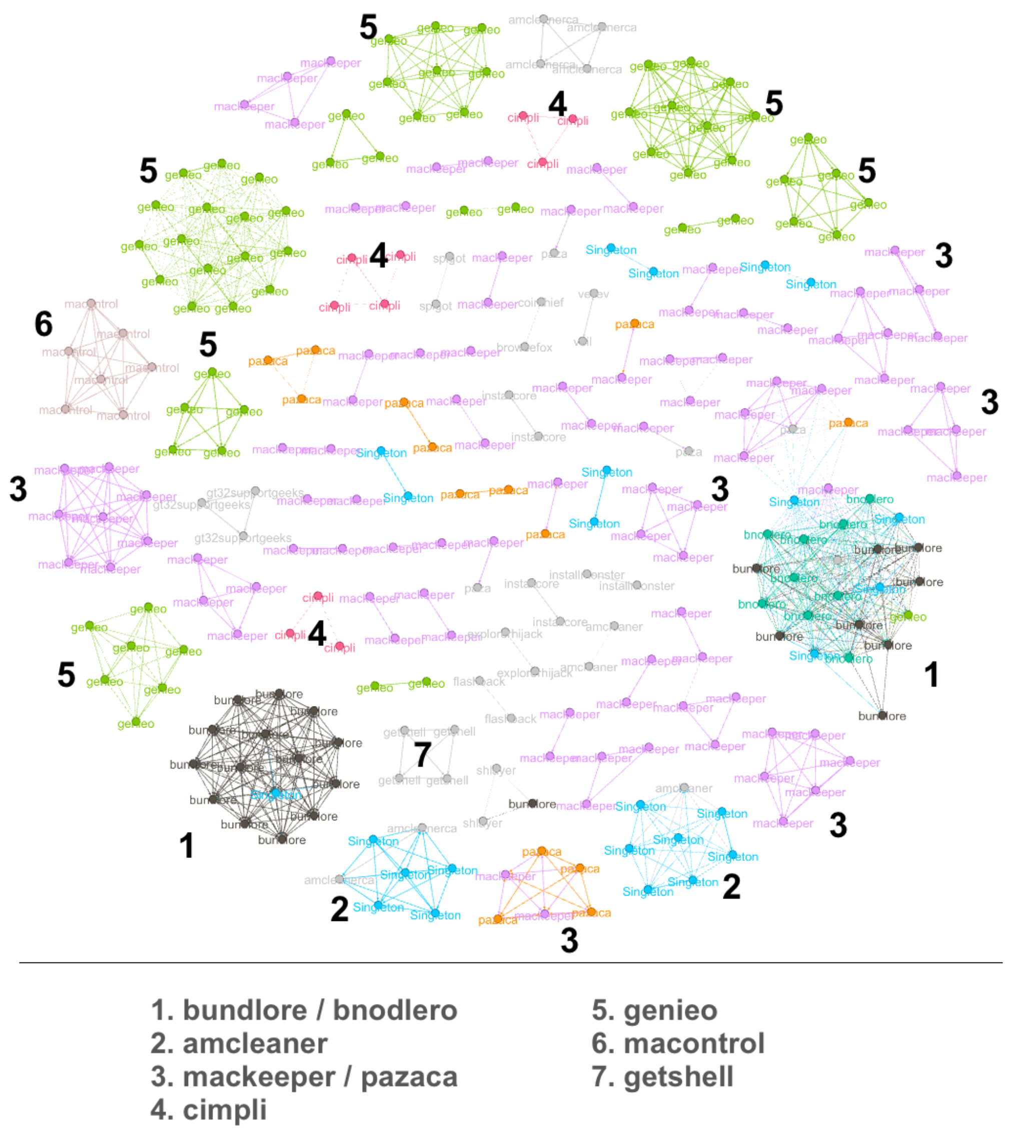


Figure 4.11: *SSDeep* clustering of observed malware

The clustering demonstrates binary similarity between samples, labelled according to *AVClass*. Seven identified clusters are labelled 1 - 7 accordingly. *Bundlore* and *bnodlero* (labelled as 1) cluster clearly in Figure 4.11 where the latter label is a mixed spelling of the former. Analysis of the underlying files observed within the clusters shows that the difference can be attributed to labelling differences that occur when a higher number of anti-malware software vendors identify a file as malicious, and use differing labelling schemes to identify them. This demonstrates that while *AVClass* makes significant headway in resolving vendor labelling inconsistency, edge cases do still occur as seen both in this case and the case of the *mackeeper* and *pazaca* (labelled as 3). This shows a close relation between these labels, and similar analysis of these files shows that both families relate to MacKeeper with inconsistent vendor labelling leading to variances in *AVClass* label assignments.

In a similar vein, manual investigation of observed file hashes of samples within *mackeeper* clusters (3) and *amcleaner* show that a number of unlabelled files (singletons) are potentially related. While out of the scope of this exploratory research, further study of labelling efficacy though use of both *AVClass* labelling and *SSDeep* clustering may provide improved methods of consistently labelling large malware datasets. Finally, and while not overly significant, a cluster of non-adware malware can be seen with the *getshell* (7) and *macontrol* (6) malware, both seen in the top ten grouping in Table 4.4.

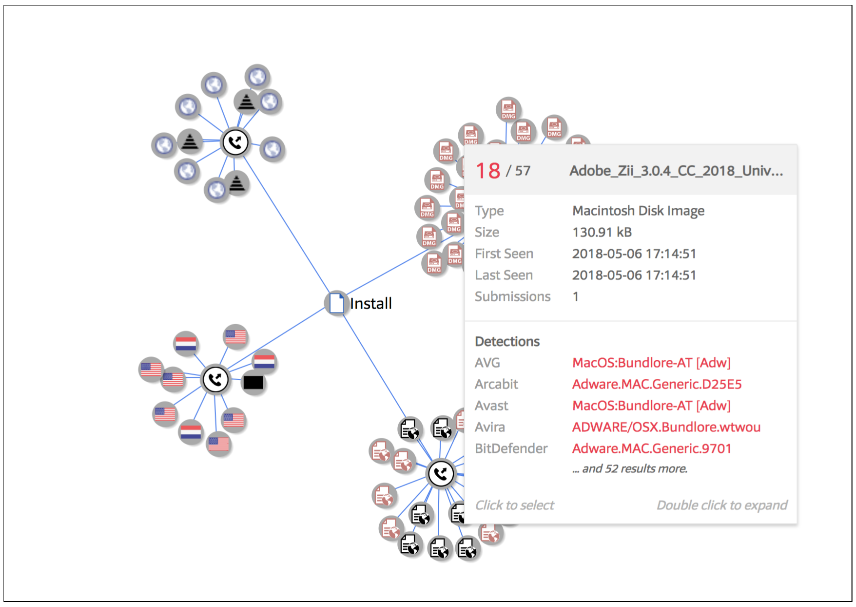
The one top 10 Category A family from Table 4.3 conspicuous in its absence is the *installcore*, however this is likely due to the reported method of installation and delivery. Malwarebytes (2018b) notes that *installcore* is frequently bundled through third party platforms as a monetisation strategy, paying distributors for successful installs and displayed adverts. This distribution through third party applications may lead to widely diverse binary payloads, with limited similarity detectable by *SSDeep*.

4.5 Infection, Persistence and Code Signing

Much of the efficacy of malware can be attributed to the combination of infection vector (the method of initial infection or installation) and the method of persistence. In addition the global growth and impact of malware has led to some operating systems, particularly Microsoft Windows, Android, macOS and iOS, implementing controls to make malicious software execution and persistence more difficult (Kotzias *et al.*, 2015; Levin, 2015). An example of this is code signing, or requiring applications to be cryptographically signed in order to install without significant user intervention, or in the case of iOS, install at all (*ibid.*). The analysis of the dataset was conducted using the same grouping used previously by looking at the top ten malware classes identified grouped into Category A and Category B, C and D. Malware within these two groups was then manually analysed and categorised in terms of infection vector, persistence mechanism and the presence and validity of any code signing. Studying these aspects provides not only insight into how these may relate to other operating systems and environments, but is also establishes a useful baseline for further studies.

4.5.1 Infection Vector

Without an initial infection mechanism, malware could not spread or be installed. Infection strategies change and adapt over time as defence strategies are updated to meet the evolving threat landscape (Ollmann, 2008; Cabaj *et al.*, 2017). To establish the most likely infection vector, this research used both the observable information in the dataset as well as the VirusTotal Graph tool[4]. The tool takes a filename or SHA as an input and shows known relationships to other observed data points within the VirusTotal database, including the filenames and filetypes identified. From this, the likely nature of the source of the infection vector could be reliably inferred. An example of this is shown in Figure 4.12, which shows a *bundlore* sample with one of the observed filenames highlighted as a fake Adobe Creative Cloud copy protection bypass file.



18 / 57 Adobe_Zii_3.0.4_CC_2018_Univ...

Type	Macintosh Disk Image
Size	130.91 kB
First Seen	2018-05-06 17:14:51
Last Seen	2018-05-06 17:14:51
Submissions	1

Detections

AVG	MacOS:Bundlore-AT [Adw]
Arcabit	Adware.MAC.Generic.D25E5
Avast	MacOS:Bundlore-AT [Adw]
Avira	ADWARE/OSX.Bundlore.wtwou
BitDefender	Adware.MAC.Generic.9701

... and 52 results more.

Click to select Double click to expand

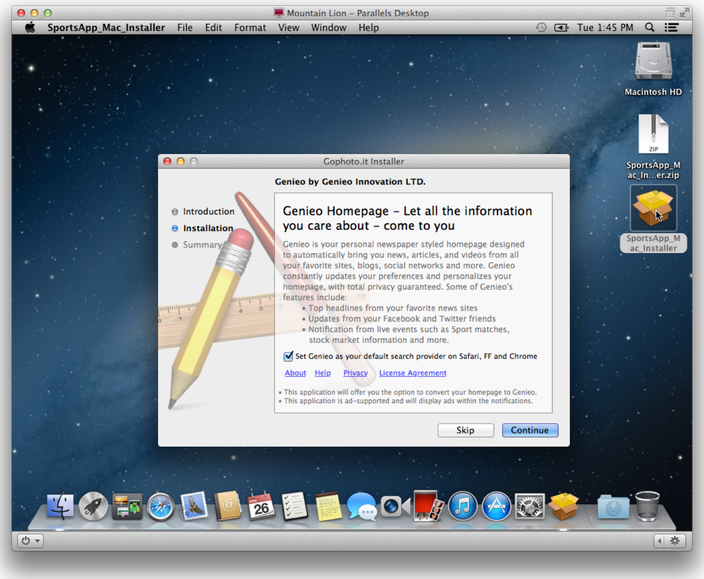
Figure 4.12: VirusTotal Graph interface, showing a *bundlore* installer source filename

The analysed infection vectors are summarised in Table 4.6 as **infection vector**. Analysis of the Category A malware shows that installation vectors for this type are either

[4]`https://www.virustotal.com/graph`

through legitimate or illegitimate software installers. Looking at the likely sources of these installers, three clear sub-categories emerge.

The first sub-category (hereafter referred to as A.1) are valid installers, where users have intentionally installed adware or PUPs, often advertised as malware removal tools (Reed, 2016b). Advanced Mac Cleaner, labelled as *amcleaner*, and the MacKeeper group of software including the *mackeeper* and *pazaca* AVClass labels are examples of this. Both claim to detect and remove malicious software, often exaggerating or falsifying detections in order to solicit ongoing subscriptions. While the practice is considered ethically and commercially dubious, the software is generally benign and removable[5]. A review of the deceptive practices used by MacKeeper can be found in Reed (2016b) and Ashenbrenner (2018).


Mountain Lion – Parallels Desktop
SportsApp_Mac_Installer File Edit Format View Window Help
Tue 1:45 PM
Macintosh HD
SportsApp_Mac_In...er.zip
SportsApp_Mac_Installer
Gophoto.it Installer
Introduction
Installation
Summary
Skip
Continue

Genieo by Genieo Innovation LTD.

Genieo Homepage – Let all the information you care about – come to you

Genieo is your personal newspaper styled homepage designed to automatically bring you news, articles, and videos from all your favorite sites, blogs, social networks and more. Genieo constantly updates your preferences and personalizes your homepage, with total privacy guaranteed. Some of Genieo's features include:

- Top headlines from your favorite news sites
- Updates from your Facebook and Twitter friends
- Notification from live events such as Sport matches, stock market information and more.

[✓] Set Genieo as your default search provider on Safari, FF and Chrome

About Help Privacy License Agreement

- This application will offer you the option to convert your homepage to Genieo.
- This application is ad-supported and will display ads within the notifications.

Figure 4.13: Genieo installation dialogue, from Malwarebytes (2013)

[5]A more detailed review of the changes in the MacKeeper product over time is given in Section 5.3

Table 4.6: Infection vectors, persistence methods and code signing: Category A

Subclass	**AVClass**	**Infection Vector**	**Persistance Method**	**Code Signing**
A.1	mackeeper	Directly Installed	Launch Agent	99% signed, of which 79% valid Consistent identifiers Consistent signers
A.1	supportgeeks	Directly installed	Launch agent	97% signed, of which 82% valid Consistent identifiers Consistent signers
A.1	amcleaner	Directly Installed	Launch Agent	99% signed, of which 98% valid Consistent identifiers Consistent signers
A.1	pazaca	Directly installed	Launch Agent	99% signed, of which 98% valid Consistent identifiers Consistent signers
A.2	installcore	Wrapper: 3rd party installer Known applications including Chrome, Sublime Filezilla and Adobe Flash	Browser extension Launch Agent	62% signed, of which 71% valid Randomly generated identifiers Varied signers
A.2	genieo	Wrapper: 3rd party installer Various generic applications and download sites	Launch agent	77% signed. of which 74% valid Randomly generated identifiers Varied signers
A.2	spigot	Wrapper: 3rd party installer Various generic applications and download sites	Launch Agent	37% signed, of which 100% valid Identifiers and signers consistent with applicable 3rd party installer
A.3	tirrip	Fake/trojan installers and pirated applications	Launch Agent	No signed executables identified
A.3	bundlore	Fake/trojan installers and pirated applications	Launch agent	33% signed, of which 91% valid Varied identifiers Varied signers
-	cimpli	Limited reference material available		

The second sub-category (hereafter referred to as A.2) are installers that bundle legitimate software, but include Category A malware as part of the payload, effectively creating a wrapper around 3rd party installers. Examples of this include *installcore*, *genieo* and *spigot*. An example of the user interface of an installer can be seen in Figure 4.13, where users are guided to install 'search helpers' while installing an unrelated tool. Software in this sub-category may have opaque removal instructions, or be otherwise challenging to remove. It should be noted that MacKeeper, mentioned in the previous sub-category, was observed by Reed (2016b) to use tactics similar to this in the form of bundling the software with a fake Adobe Flash updater. This was not, however, observed in the dataset under analysis. *Installcore* is notable for bundling the adware payload around seemingly valid software installs of popular software packages, including Google's Chrome browser, Filezilla, the Sublime text editor and Adobe Flash as seen in both this dataset and the work of Abbati (2015). In an investigation of macOS cryptomining malware, Reed (2018c) discusses the role of file aggregation and download sites in distributing this style of malware, noting "such sites have a long history of issues" (p. 2) and calling out *macupdate.com* in particular as distributing third party software along with adware installers during 2015.

Finally, the third sub-category (hereafter referred to as A.3) as Category A malware are installers that use highly deceptive tactics to trick users into installing them. Analysis of the installation sources shows that many of these are distributed as fake software updates and installers, effectively operating as 'trojan horse' style malware. In their analysis of the *pirrit* malware (classed by AVClass as *tirrip*). Serper (2016) notes that not only is the software distributed as fake updates and pirated copies of common software packages, but that the installation packages use a pre-install script to load a dropper responsible for pulling down additional malicious payloads before a user has provided any input or elected to continue with the installation process. The methods of infection used by this sub-class are far closer to the methods seen in non-Category A malware, discussed next.

Table 4.7 summarises the analysis of the top ten Category B, C and D malware. While the sample size is very small in comparison to Category A malware, the methods of infection show variance in comparison. *Getshell*, the most frequently observed non-Category A malware, was distributed through a fake Adobe Flash updater on a compromised website, a similar infection vector to the more malicious examples of Category A malware (F-Secure, 2012b). A notable infection vector can be seen with *keranger* and *proton*, both of which compromised popular macOS software to install without the users knowledge: in the case of the *keranger* ransomware, the popular *Transmission*[6] bitorrent client, and

[6]`https://transmissionbt.com`

the *proton* RAT through media tools *Handbrake*[7] and *Elmedia*[8]. This method of supply chain attack, compromising software assumed to be secure by packaging malware and distributing it as a valid version, has proven a popular route for malicious actors (Xiao and Chen, 2016). An example of the infection vector being used successfully can be seen in the case of Panic Software. The user in the company was infected by *proton* through the *Handbrake* compromise, suffered data loss as a result and as developers of popular macOS software themselves, opened up the possibility of similar supply chain attacks using their own code base at a later date (Frank, 2017).

4.5.2 Methods of Persistance

Just as malware with no infection vector cannot be effective, the ability of malware to persist is equally important. Wardle (2014) notes that “persistence is essential for malware” (p. 16), and provides a comprehensive breakdown of the common methods of persistence of macOS malware. Furthermore, while infection vectors could be shared across operating systems (Bestuzhev, 2018), persistence mechanisms are generally unique to the operating system the malware is targeting as the ability to persist a system restart would require use of operating or platform specific functions.

While Wardle (2014) identified in excess of 13 different techniques and methods malware could use in order to persist on macOS, only 3 of these are identified within the two top ten datasets, detailed in Tables 4.6 and 4.7. Within Category A, only launch agents and installed browser extensions were observed. Within Category B, C and D a greater variety of persistence methods were observed, with launch agents used along with kernel extensions (in the case of *rubilyn*) and a simple process sleep for the *keranger* ransomware, which waits for three days before starting the file encryption process.

[7]`https://handbrake.fr`

[8]`https://mac.eltima.com/media-player.html`

Table 4.7: Infection vectors, persistence methods and code signing: Categories B, C and D

AVClass	Infection Vector	Persistance Methods	Code Signing
getshell	Compromised website delivered fake/trojan installer	Various; secondary payload downloaded on execution	None
keranger	Included in compromised installer of popular bitorrent client	Ransomware, executed process sleeps for 3 days before execution	Valid, included with signed 3rd party installer
macontrol	Email with document attachment containing malicious payload	Launch Agent	None
macransom	Malicious executable	Launch Agent	None
proton	Included in compromised installer of popular video tool	Launch Agent	Valid, included with signed 3rd party installer
rubilyn	Malicious executable	Kernel Extension	None
SINGLETON:48288 (OSX.Miner)	Limited reference material available; derived from valid coin mining tool		
SINGLETON:72c2f (Uknown)	Limited reference material available		
SINGLETON:8ab73 (Mac.CoinMiner)	Limited reference material available; derived from valid coin mining too		
SINGLETON:b7b69 (Trojan.WisdomEyes)	Win32 Malware identified on macOS machine, likely from external drive		

4.5.3 Code Signing

Code signing allows for developers and distributors of software to cryptographically sign software, allowing the end users operating system to validate the authenticity of software before installing. In most cases[9], any unauthorised modifications to a package or executable would render the signature invalid thus generating a warning or preventing execution (Alrawi and Mohaisen, 2016). The macOS operating system was an early adopter of code signing as a control, with implementations in OS X version 10.5 in 2007 initially as an added (but non-mandatory) security feature eventually becoming firm guidance in OS X version 10.8 in 2012 via the Gatekeeper system (Levin, 2015). The absence of a valid signature does not stop software from being executed, however it has required progressively more warnings and intentional action for users to bypass the control should they wish to install unsigned software.

To support this control, developers have been required to sign packages to prevent warnings, and as a mandatory requirement to release them within Apple's App Store ecosystem. Both the installation packages as well as individual Mach-O executables can be signed, and software developers wishing to release signed software packages are vetted by Apple and have to pay an annual fee in order to qualify. At first glance, this would appear to create a reasonably difficult barrier for malware authors on the macOS platform, but analysis of the data shows otherwise.

The **code signing** column of Tables 4.6 and 4.7 provide a summary of the observed signatures of the top ten classes in both the Category A and Category B, C and D groups. Signature information was identified through metadata provided by VirusTotal for each unique observed hash in the two groups, and manually reviewed to identify patterns. Once identified, the percentage of signed executables was noted, and the corresponding percentage of valid signatures within that class recorded. Additionally, where observed the package identifiers[10] were analysed for validity and consistency.

Manual analysis of the Category A malware shows that the three sub-classifications identified as part of the analysis of infection vectors (A.1, A.2 and A.3) are supported when considering the prevalence of signatures, the validity of the signatures observed and the

[9]Amongst other design issues, the frequency of validity checks are infrequent which may lead to vulnerabilities (Reed, 2018a)

[10]The package identifier or Uniform Type Identifier (UTI) is an internal identifier using a 'reverse DNS' notation, usually in the form *com.domain.text*, used to distinguish different applications or application bundles. More details can be found at `https://developer.apple.com`

Table 4.8: Examples of Category A.1 malware

AVClass	Valid Certificate	Signed	Package Identifier / Signed By
mackeeper	TRUE	TRUE	com.mackeeper.MacKeeper.plugin.DuplicateFinder KROMTECH ALLIANCE CORP.
mackeeper	TRUE	TRUE	com.mackeeper.MacKeeper KROMTECH ALLIANCE CORP.
mackeeper	FALSE	TRUE	com.mackeeper.CerberusInstaller KROMTECH ALLIANCE CORP.
mackeeper	FALSE	TRUE	com.mackeeper.MacKeeper.MKCleanService KROMTECH ALLIANCE CORP.
amcleaner	TRUE	TRUE	com.techyutils.cleaner Techyutils Software Private Limited
amcleaner	TRUE	TRUE	com.pcv.rlistupdater Techyutils Software Private Limited
amcleaner	TRUE	TRUE	com.techy.Mac-Optimizer Techyutils Software Private Limited

use of meaningful package identifiers. Category A.1 malware shows consistent code signing and meaningful package identifiers, a sample of which can be seen in Table 4.8 from the *mackeeper* and *amcleaner* classes. Table 4.6 shows that *mackeeper*, *amcleaner*, *pazaca* and *supportgeeks* all show consistent numbers of signed files (90% or higher), of which a high percentage are valid certificates (70% or higher).

A sample of category A.2 malware can be seen in Table 4.9. Where signed, package identifiers appear to be randomly generated and the signatories appear to use a variety of names, with few exceptions. Table 4.6 show that category A.2 has a lower number of overall observations signed, with *genio* having 77% signed, of which 74% were found to be valid and *installcore* with 62% signed of which 71% were valid.

Finally, category A.3 malware samples can be seen in Table 4.10. The two classes in the category have a very low number of signed executables: none in the case of *tirrip* and 33% in the case of textitbundlore, of which 91% were valid. Where package names are present in signed executables, the names appear to be incongruent with the packages themselves and are likely meaningless.

A summary of the code signing details of Category B, D and D malware is included in

Table 4.9: Examples of Category A.2 malware

AVClass	Valid Certificate	Signed	Package Identifier / Signed By
installcore	TRUE	TRUE	com.hellkite.jester Mark Lloyd
installcore	TRUE	TRUE	com.profaculty.netwise Ran Greenberg
installcore	TRUE	TRUE	com.kandol.Ootocoidea Andy OS, Inc.
genieo	TRUE	TRUE	com.racol.AppRC Rafi Colman
genieo	FALSE	TRUE	com.papp.PApp Zahi Shanhav
genieo	FALSE	TRUE	com.davidkadi.AppDKI David Kadishi

Table 4.10: Examples of Category A.3 malware

AVClass	Valid Certificate	Signed	Package Identifier	Signed By
bundlore	TRUE	TRUE	com.stubberify.mym	Oleg Krokhinov
bundlore	TRUE	TRUE	com.tostubornot.mym	Sergey Vasiliev
bundlore	TRUE	TRUE	com.macdownloader.apple	Olga Asadcheva
tirrip	all samples have no certificate or package identifiers			

Table 4.4. In most cases, executables were not signed. The two notable exceptions were the *keranger* and *proton*, both of which used valid certificates of compromised clients used as the infection vector for the malware. While the low sample size makes extrapolation of meaningful patterns impractical, the relative success of the *proton* and *keranger* malware in practice indicate that code signing as a security strategy can be bypassed through similar strategies (Reed, 2018b). In their work analysing code signing practices of Windows malware Kotzias *et al.* (2015) found that adware and PUPs were often signed and other categories of malware rarely so, a finding supported by this analysis.

4.6 Anti-malware software vendor consensus

Different anti-malware software vendors use different methods of detection to establish the likelihood of software being malicious. Methods may include static, dynamic and statistical analysis that use known signatures, suspicious behaviour and machine learning techniques to identify potential malware as quickly and consistently as possible (Damodaran *et al.*, 2017). With the observations having been enriched with data from VirusTotal, information on vendor consensus is available as *positives*, discussed in Section 3.7. Wardle (2018c) discusses multi-vendor detection on the VirusTotal platform, and notes that the detection engines running within the VirusTotal sandbox environment do not always match those of end user systems, however an analysis of the number of recorded positives does provide insight into the general consensus between vendors that observations are malicious.

Figure 4.14 shows a histogram of the number of vendors rating various malware samples as positive for Category A malware. A normal distribution is observed, with samples being detected by between one and five vendors showing a small increase outside the norm, which may be caused by newer samples not yet being recognised as malicious. The majority of samples show a positive identification of between fifteen and thirty vendors. Figure 4.14 shows the Category B, C and D malware, and reflects the lower overall size of the dataset. A more signifcant number of the malware detected was by between one and five vendors (eleven), versus the forty seven types malware detected that can be seen in the central distribution, ranging between twenty five and forty vendors as the lower and upper bound respectively.

To investigate this further, the correlation between this vendor consensus and the first time a type of malware was observed was explored. An additional data point made

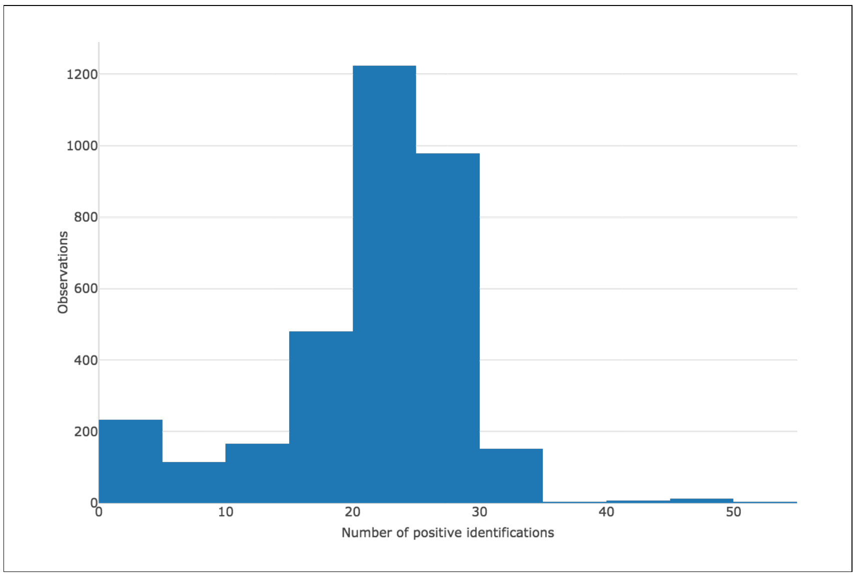


Figure 4.14: Histogram of positives: Category A

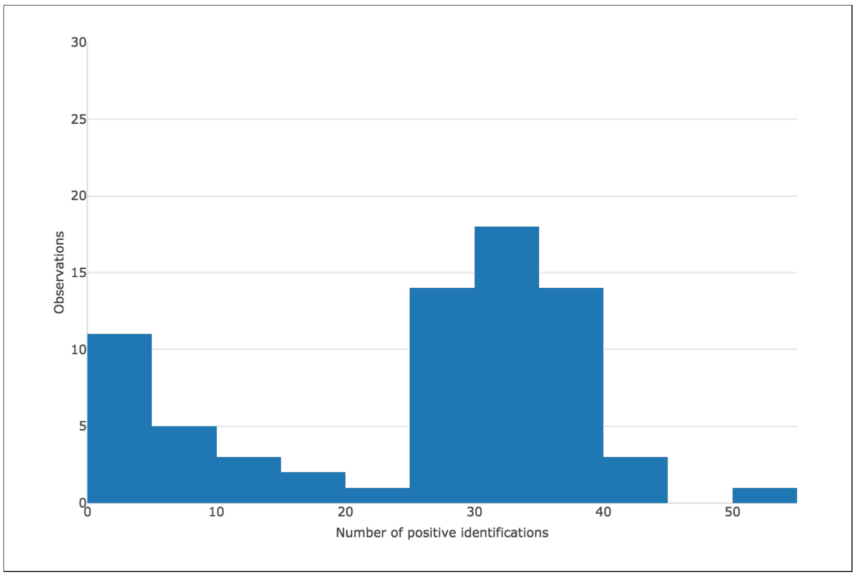


Figure 4.15: Histogram of positives: Category B, C and D

available by the enrichment of the data with VirusTotal information is the number of unique sources a sample has been submitted by. This figure provides an indication of how wide spread a particular malware sample may be. Visualising this allows the research to examine if the age of malware influences the number of vendor observations, as may be intuitively assumed. Further to this, by sizing the data point by the number of unique observations, the relative spread of the malware in the wild can be inferred.

Figure 4.16 shows the vendor consensus plotted against the date a malware sample was first seen for Category A malware, using the number of unique sources for size. From the plot, the concentration of malware shows that observed malware recent malware clustered between a vendor consensus of around eleven and thirty vendors with the majority of malware first seen relatively recently (after 2015). Generally the number of unique sources can be seen to be reasonably consistent at five or below. This can be summarised as following: most of the Category A malware observed come from around five different sources, are generally detected by between eleven and thirty different anti-malware software vendors and have been seen in the wild since around 2015, with a concentration in recent years (2017 and 2018).

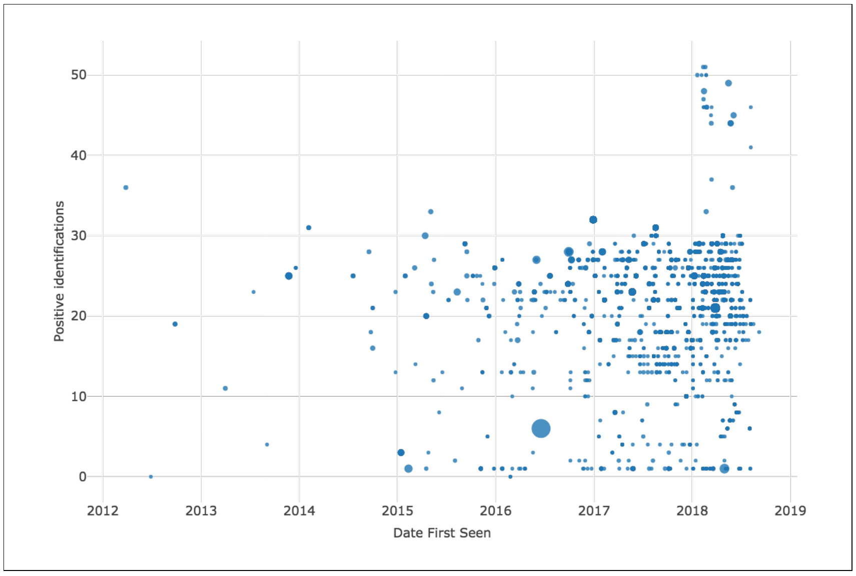


Figure 4.16: Vendor consensus and first seen time sized by unique sources: Category A

Building on this analysis, a second visualisation was carried out where the sub-categories detailed in Table 4.6 were assigned sub-categoriesA1, A.2 and A.3 respectively. The points for each sub-category were assigned a different color to assist with visualisation. A clear grouping is observed between sub-category A.1, highlighted in box 'A.1', recognised as malicious by between approximately 13 and 23 vendors, and the more malicious sub-categories A.2 and A.3, highlighted in box 'A.2 and A.3' between approximately 22 and 30 vendors. This finding indicates that vendor consensus appears to increase in the case of more malicious Category A malware.

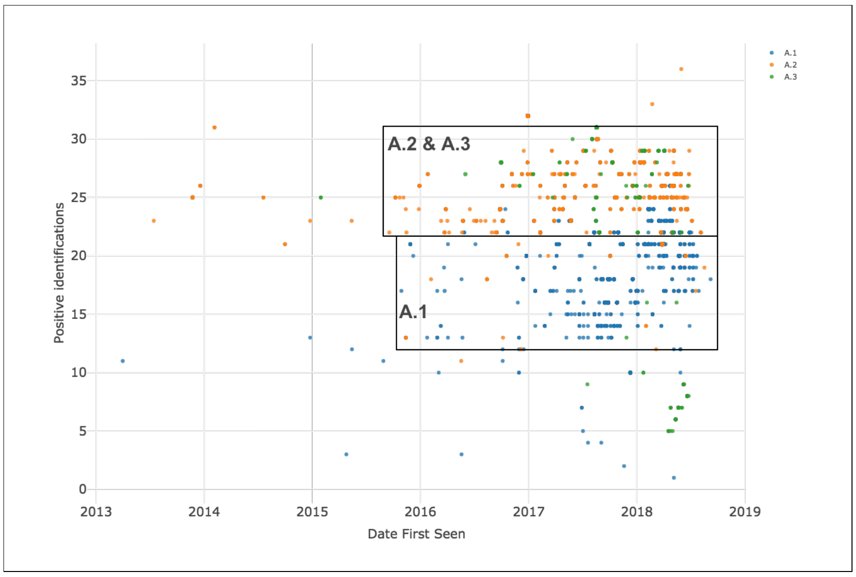


Figure 4.17: Vendor consensus and first seen time: Category A.1, A.2 and A.3

Figure 4.18 provides the same visualisation as above for Category B, C and D malware. The significantly smaller number of observations is evident, however from those plotted two differences are apparent. The first difference is that the spread of malware indicated by unique sources is higher, with a number of observations recorded from more than ten unique sources. The overall consensus level for the categories of malware are slightly higher, with most samples falling between twenty and forty vendors. The malware detected by a lower number of vendors (between 1 and 5 vendors, highlighted in Figure 4.15 on the left side of the distribution) are more recently seen malware. In this category, the plot can be summarised as showing a higher proportion of the Category B, C and D malware observed come from a more unique sources (ten or more) than Category A, are generally detected by a larger number of anti-malware software vendors (between twenty and forty) and have been seen in the wild for a slightly longer period (2012 onwards).

4.7 Summary

In this section the cleaned, aggregated and enriched dataset was analysed using an exploratory approach. Analysis included a temporal analysis of various aspects of the data,

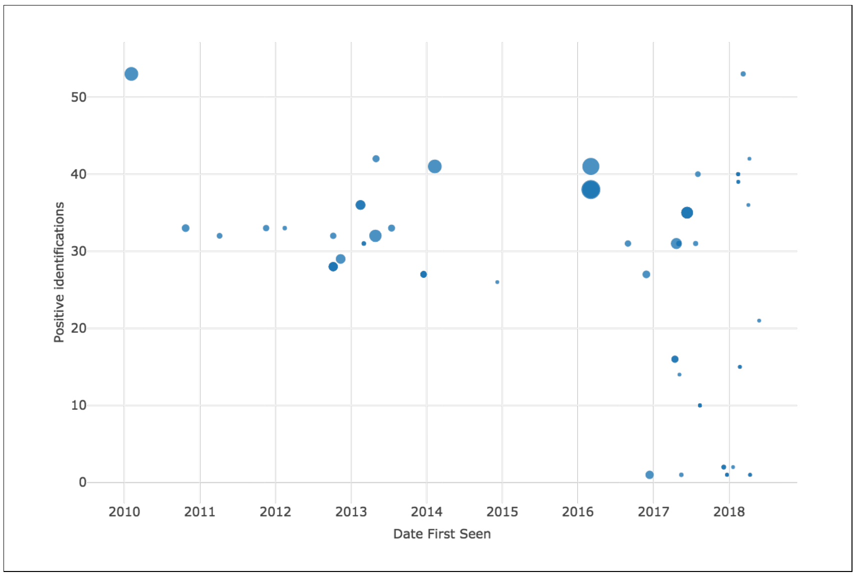


Figure 4.18: Vendor consensus and first seen time sized by unique sources: Category B, C and D

providing insight into the observations over the full eleven month period of the dataset, as well as weekly and daily breakdowns. The major categories and identified classes of malware were analysed and ranked accordingly, and the dataset further refined into two significant groups: Category A malware, consisting of adware and PUPs and Category B, C and D malware consisting of RATs, ransomware and coin miners.

Building on the analysis of categories and classes within the dataset, the diversity of the population was considered, showing an increase over the dataset time period. Additionally, clustering of observed malware using established work on malware clustering allowed grouping and visualisation of the dataset. Finally, the infection vectors, persistence mechanisms and code signing was analysed, identifying three sub-groupings of Category A malware based on these factors: Categories A.1, A.2 and A.3.

Consensus in detection of malware by multiple vendors was also explored in the dataset, showing that Category A malware enjoys slightly lower consensus beteeen vendors. Within the category, however, further support is found for the sub-groupings previously identified. Additionally, Category A in seen in higher concentrations in recent years, specifically from 2015 onwards. The distribution of the malware measured by unique sources is also lower in Category A compared to Categories B, C and D.

In the chapter that follows, the results will be discussed with a view to building a model for additional work in the future, a better understanding of the macOS malware landscape and insights that may be useful in practice.

Chapter 5

Results and Discussion

"Inconceivable!"
"You keep using that word. I do not think it means what you think it means."

William Goldman, The Princess Bride

5.1 Introduction

Industry reports, user perception and scholarly research have been at odds on the nature of malware on macOS. On one side, users view the operating system as more secure and less prone to malicious activity (Wash, 2010; Howe *et al.*, 2012). On the other, practice, press and security vendors have been predicting everything from a steady increase to veritable tsunami of macOS malware since the *FlashBack* trojan appeared in 2011 (Manning, 2012; Kaspersky, 2018; Symantec, 2018a). In the middle, the research community has justifiably focused on platforms with significant usage footprints and availability of data, focusing efforts on Windows and Android malware ecosystems (for example, Soto-Valero and González (2018) and Yen *et al.* (2014)). This research aimed to provide some clarity by filling a gap in empirical studies of macOS malware, hypothesising that macOS malware does represent a credible threat to security on the platform, and that an understanding of the nature of this malware could be applied in practice to improve the overall security posture of users and administrators alike.

The research questions posed in this work looked to test the hypothesis by reviewing related work on the topic of malware as a whole and macOS in particular. This was to be augmented by exploring a a recent dataset of macOS malware observations in a real-world environment to answer ‘what’, ‘how’ and ’when’ questions. In the sections that follow, points from related work will be integrated with the findings from the exploratory data analysis to understand these results, answer the research questions and understand the implications of the results. Following this limitations of the research will be discussed, and finally recommendations for future work will be considered.

5.2 Integrating Existing Work

In the review of related work in Chapter 2, the current malware ecosystem was explored. While all the areas covered provide background, context and work related to this research, three key areas stand out when evaluating the results.

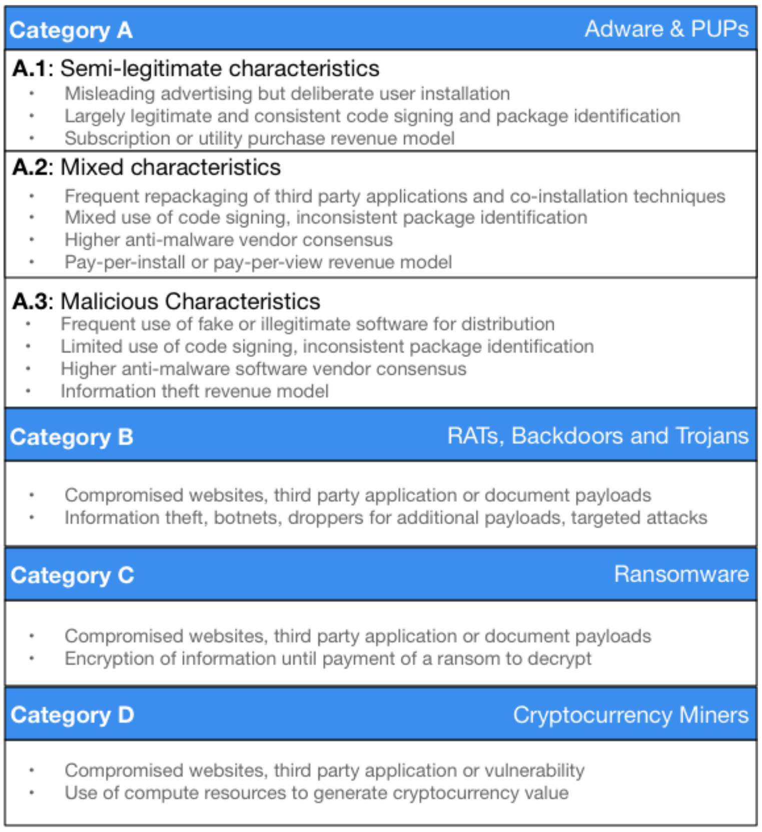
5.2.1 Methods of infection and persistence

The first area is the methods of infection and persistence discussed in Section 2.3.2 and analysed further in Section 4.5. While a number of methods of infection are identified in related work, analysis shows that a limited set of these are seen in the malware observations. For the most part, Category A malware makes use of either direct installation, or wrappers around other applications. Category B, C and D malware is more varied, including compromised websites and compromised third party installers. This shows that while a number of methods of infection could be taken advantage of, only a small number appear to be actively used. Similarly, numerous persistence mechanisms are identified yet a limited number are observed: generally, standard methods of macOS persistence through launch agents are used, with the occasional use of kernel extensions and browser based persistence. Diversity of persistence mechanisms is much higher in Category B, C and D malware, again suggesting that while a number of methods exist to successfully persist malware, few are observed in the wild.

5.2.2 Naming, labelling and clustering

The second area considers the challenges of naming, labelling and clustering of malware, discussed in Section 2.4 and analysed in Section 4.3 and Section 4.4. *AVClass* has been used to label of large malware datasets, and was successfully applied in this research to provide normalised labels for the observed malware. Additionally, *SSDeep* was utilised to cluster malware, revealing that a combination of both automated labelling with AV-Class and automated clustering with SSDeep provided a high level of consistency and identification of aliases.

Categories A, B, C and D were defined by considering a review of practice literature and reports and reviewing high level findings in the observed malware dataset, and are shown in Figure 5.1. The top ten observed families where identified and categorised, with Category A making up a significant majority of the malware observed. The balance, Categories B, C and D, comprised a minority. An important finding of this research was that when Category A was explored in greater detail, three sub-categories emerged: Category A.1 comprised of close-to-legitimate malware, Category A.2 comprised of malware that showed a mix of legitimate and illegitimate characteristics and Category A.3 comprised of malware that demonstrated almost no legitimate aspects, and shared a number of characteristics with non-Category A malware. Section 5.1 outlines the high level characteristics based on the findings from this research.



Category A Adware & PUPs

A.1: Semi-legitimate characteristics
- Misleading advertising but deliberate user installation
- Largely legitimate and consistent code signing and package identification
- Subscription or utility purchase revenue model

A.2: Mixed characteristics
- Frequent repackaging of third party applications and co-installation techniques
- Mixed use of code signing, inconsistent package identification
- Higher anti-malware vendor consensus
- Pay-per-install or pay-per-view revenue model

A.3: Malicious Characteristics
- Frequent use of fake or illegitimate software for distribution
- Limited use of code signing, inconsistent package identification
- Higher anti-malware software vendor consensus
- Information theft revenue model

Category B RATs, Backdoors and Trojans
- Compromised websites, third party application or document payloads
- Information theft, botnets, droppers for additional payloads, targeted attacks

Category C Ransomware
- Compromised websites, third party application or document payloads
- Encryption of information until payment of a ransom to decrypt

Category D Cryptocurrency Miners
- Compromised websites, third party application or vulnerability
- Use of compute resources to generate cryptocurrency value

Figure 5.1: Overview of observed malware categories and their characteristics

Outside of the scope of this study and discussed in further opportunities for research is the integration and synthesis of existing taxonomies with this finding, and identifying potential subcategories in other types of malware not observed in this dataset.

5.2.3 macOS perception of security

The third area discussed in Section 2.5 focuses on the macOS operating system, and highlights that while user perception of the operating system may be one of enhanced security, a review of the known vulnerabilities shows that the system has in certain years have seen similar (and in some cases, more) published vulnerabilities than that of its Microsoft Windows counterpart. Working purely with public available data such as the National Vulnerability Database as a proxy for overall operating system security should be approached with caution, however when considered alongside the relative severity of the identified vulnerabilities, the underlying notion of the operating system being ‘more secure by default’ can be challenged. From this and the known infection and persistence mechanisms, it suggests that the lack of diversity in macOS malware has more to do with market share and economic factors and lower likelihood of reward than superior security controls.

5.3 Exploratory Analysis

The real world dataset was ingested, aggregated, classified, labelled and clustered. Analysis of the data focused on temporal analysis (the ‘when’), and the nature of the observed malware itself (the ’what’ and ’how’) in terms of classification, diversity, infection vectors, persistence and code signing. As this was exploratory analysis, the dataset was approached as far as possible with no preconceived outcomes: the data, and any points of interest, was explored to build a view in such a way that it could be repeated on different datasets, or for different operating systems, for comparative analysis. The key areas and their findings are discussed as follows.

5.3.1 Temporal Analysis

Temporal analysis revealed observations broadly in line with industry reports. In analysing this, a number of factors should be considered. Firstly, it is unknown what impact the growth of the anti-malware vendor client base may have had on the increased number of detections as client numbers where not available to this researcher, and are considered commercially sensitive information. On the assumption that these numbers were relatively static, an increase is observed in the period under observation in both Category A and Category B. Industry studies (Kaspersky, 2018; Symantec, 2018a) report a growth in malware in categories C and D, but this is not reflected in the dataset. This indicates that while risks are present, the increase appears to be in traditional adware, PUPs, trojans and backdoors rather than newer types of malware like ransomware and cryptocurrency miners. Further opportunities exist to explore different real-world datasets for similar patterns, and correlation to specific geographies.

5.3.2 Diversity

The lack of diversity in observed malware may underpin the perception that malware is less of a risk on macOS. Analysis of the data does not strongly support an increase in diversity of malware over time, as seen in the Android study by Soto-Valero and González (2018), although Category B, C and D malware do show an increase in diversity over the period (albeit with a small dataset). Although not considered in this research, additional research may measure and analyse factors impacting diversity in observed datasets, or comprehensively explore what ratio of known macOS malware datasets like Wardle (2018a) can be seen in real-world datasets.

5.3.3 Infection Vectors and Code Signing

In the analysis of the infection vectors, methods of persistence and code signing observations were considered within the top ten malware in Category A, and Category B,C and D. Analysis of these characteristics contributed further to the sub-categorisation of Category A malware seen in Figure 5.1. Limited variation was observed in methods of persistence, and the most common infection vectors were direct installation and via third party installers. When considered in the context of direct infection through vulnerabilities, there was no evidence in this dataset to support this as a common method of infection on macOS at this point. This implies that for the most part, users directly authorise installation of the malware rather than the malware requiring exploitation of a vulnerability to infect the user. As discussed earlier, this does not imply an immunity to such vectors, simply that the method of infection is currently not commonly seen. Analysis of code signing provided insight into the techniques used, especially by Category A malware authors, and confirmed related work discussed in Section 2.3.3 that code signing is far from infallible, and can be bypassed regardless of commercial and verification controls.

While not directly reflected in the analysis, the prevalence of MacKeeper and its relative legitimacy was of interest to this researcher as the malware dominates the observed dataset, but presents as legitimate software when looking at the characteristics studied in this research. Specifically, the infection vector is direct, in that users choose to install the software, methods of persistence are routine and the code signing legitimate with valid package identifiers. A number of anti-malware vendors consider the software to be malicious, and Reed (2016b) provided technical reasons why, in 2016, this may have been the case. Later versions of MacKeeper appear to have addressed a number of the criticisms levelled in practice, although not all (Ashenbrenner, 2018). In a lab environment as part of this exploratory research, the product was downloaded, installed and tested and shown to be straightforward to install and uninstall, with no false positive detections recorded. While this is not intended to be a comprehensive review of the product, what is of interest is that a Category A adware product may have, at least in part, legitimised over time. This may be related to ongoing detection by anti-malware sofware vendors, and may provide a potential case study outside of the scope of this research but worthy of further investigation.

5.3.4 Vendor Consensus

Finally, anti-malware software vendors demonstrated consensus for a large portion of the observed macOS malware dataset. Hurier *et al.* (2016) discusses the lack of consensus between anti-malware software vendors by evaluating existing ground truth databases against a large body of malware from VirusTotal. In this research, malware from the dataset was plotted using the date a malware sample was first seen, the number of positives (i.e., ratings by an anti-malware software vendor as malware) and the number of unique sources submitting a malware sample. Malware identified by fewer vendors was more noticeable in Category A malware, and malware from a greater number of unique sources was shown with Category B, C and D malware.

Distinguishing between malware identified in sub-categories of Category A demonstrated that groupings could be visually identified when plotted in the same way. Sub-category A.1 displayed lower levels of vendor consensus than A.2 and A.3, which may be as a result of the semi-legitimate characteristics of the malware, as opposed to A.2 and A.3

with limited or no legitimate characteristics. This difference in vendor consensus between categories further supports the sub-categorisation of Category A malware.

5.4 Implications and Future work

Sections 5.2 and 5.3 discussed the analysis in order to answer what malware families have been observed on the macOS platform, how much diversity is present in the population of observed macOS malware and how the detection rates have increased over time. Additional outcomes of the exploratory analysis have also been discussed. In the next section, the final research question will be considered: the implications and direction of future research that should be taken to understand the factors influencing malware on macOS.

5.4.1 Implications

A key finding and implication of the research is that the volume of malware observed on macOS while increasing over time is not overly diverse, and that at first glance the type of malware may appear to be reasonably benign adware. This research has shown that in fact not all adware is equal, and many of the observed families represent a credible threat to both privacy and security. Technical analysis has shown that the underlying operating system is no more secure than its counterparts in terms of catalogued vulnerabilities, infection vectors and methods of persistence. Additionally, operating system controls such as code signing may be ineffective deterrents for motivated malware authors. Ultimately, these authors have the ability to write and spread costly and damaging malware, but possibly not the economic incentive. This has practical implications for those using and administering macOS environments - an increased adherence to security best practice is encouraged, regardless of any perception of security that may exist.

Thus, malware has been shown to be a credible threat on the macOS platform. The groundwork and ability exists for real threats from malware to increase, and should other influencing factors change, such as an increase in market share or perceived value of macOS users, the diversity of and damage caused by macOS malware may well increase.

5.4.2 Future Work

The direction of future research lies in additional studies, from different real-world datasets, that offer the opportunity to validate or improve on the findings of this work. Work that is able to compare and contrast observations from multiple operating systems would bring significant value to the field of macOS malware research. Longitudinal studies of macOS malware corpora over a longer periods similar to Yen *et al.* (2014) and Soto-Valero and González (2018) would also be beneficial.

In terms of labelling, categorisation and taxonomies, there exists an opportunity to further explore methods of labelling large datasets that use both vendor label analysis and normalisation as well as automated clustering. Application of one or more of the many existing taxonomical approaches to malware may bring additional clarity to macOS malware, including the categorisations proposed in this study. Additional work that can validate the sub-categories identified would be valuable in both research and practice.

Finally, existing investigations into the efficacy of code signing of malware both on the macOS platform and in a broader context show promising results, and could be extended into a more detailed review to establish ways that code signing code be improved by operating system vendors, taking into account the current lack of effectiveness of financial barriers and validation efforts.

5.5 Limitations

Research is conducted subject to limitations and bias, and this research is no exception. The use of a single vendor dataset may be considered limitation, as the data is limited by the reach of the product and the geographies it is used in. The anti-malware software product that provided the malware telemetry is focused on the corporate user market, so these finding may no reflect individual, or non-corporate users usage patterns. Additionally, the relatively short eleven month window the data covers is may be a limiting factor.

In terms of the technical nature of malware observations, it should be highlighted that the dataset comprises only of malware that was detected by the product in use. In many of the cited industry studies, and especially those that may consider browser based malware with no physical artifacts (for example, certain cryptocoin miners) on a host as prevelant malware, these may not be detected and thus not included in the dataset under analysis. An additional limitation related to the composition of the dataset is the anonymity of users, which may introduce a risk of invalid data (for example, test data) being included. This is a potential limitation which was mitigated as far as possible and discussed in Section 3.9, but may still occur.

Finally, the overall nature of a dataset of malware observations from an anti-malware software vendor is subject to a natural survivorship bias; malware infections rendering an endpoint unusable or disabling anti-malware software would not be visible in the dataset.

5.6 Summary

This Chapter considered the research questions and proposed hypothesis, integrating observations from related work with the results of the analysis carried out. The implications of the research were discussed in Section 5.4.1, and possible future work suggested based on the results. Limitations of the research were also discussed. In the chapter that follows, the work will be summarised and concluded.

Chapter 6

Conclusion

Adware is just malware with a legal department

Amit Serper, Virus Bulletin Conference, 2018

The study of malicious software, or malware, has developed significantly as a research area over the last four decades. From Cohen's early use of the term 'computer virus' to modern work spanning prevention, detection and forensic work, both scholarly and practical aspects of the field have grown in scope and and depth.

This research examined the history of malware as a threat to users and administrators alike. Existing literature and background information was reviewed in Chapter 2, including work on detection, analysis and evolution of malware, common terminology, infection mechanisms and methods of persistence. Specific attention was paid to operating system controls like code signing, and extant literature and works in the areas of malware classification, labelling, clustering and taxonomies. The relative security of the macOS operating system was explored using publicly available vulnerability datasets, finding that the perception of macOS being a more security operating system was not always supported: macOS shares a comparable level of vulnerabilities, and similar mechanisms of control to its more widely used desktop operating system counterparts.

This research hypothesised that malware on macOS was a credible threat to users of the platform. To test this, exploratory research was undertaken on a dataset of real-world malware observations on macOS endpoints. With both testing on real world datasets and providing sufficient details of experiments undertaken suggested as prudent practice for malware analysis by Rossow *et al.* (2012), Chapter 3 discusses the methodology used in the research in detail. Data was imported from an anti-malware software vendor provided dataset for an eleven month period, aggregated, enriched with data from VirusTotal and categorised using both *AVClass* and a custom categorisation method to establish high level categories. In Chapter 4, the resulting dataset of 3,450 observations were analysed. The objective of the analysis was to provide detail of the nature of malware on the macOS operating system. This included temporal analysis, a study of the outcomes of labelling, classification and clustering and the diversity of observed malware in the dataset. A review of the observed persistence methods, infection vectors and code signing was also undertaken, and insight into consensus between anti-malware software vendors was gained.

A number of conclusions were taken from the work carried out, and were discussed in Chapter 5 and summarised here. Malware is a credible threat to users of the macOS operating system. The nature of the malware in an observed data set does not reflect the findings of industry reports; cryptocoin mining and ransomware are infrequently observed compared to the number of adware and PUPs seen. However, when the latter category of malware is explored in detail, this research shows that three clear subcategories emerge and that the characteristics of these categories range from somewhat innocuous to highly malicious. The operating system controls in place to control the spread of malware can be seen to be bypassed through technical and commercial means, and overall malware can be seen to be growing. Finally, the limitations of research and suggestion directions for future work were discussed.

In light of this, it can be concluded that the threat of malware on the macOS platform may not always take the shape expected, but presents a valid threat for users of the platform, and the hypothesis is supported. Considering the relatively small share of the market that macOS enjoys, diversity and threats are only likely to increase along with this market share. From a practitioner standpoint, attention should continue to be paid to following good security practice, usage of available anti-malware software, ensuring vulnerabilities are patched and ongoing user education.

References

Abbati, A. OSX.IronCore.A, or what we know about OSX.FlashImitator.A. Website, February 2015. `https://www.sentinelone.com/blog/osx-ironcore-a-or-what-we-know-about-osx-flashimitator-a/`, accessed December 2018.

Alperovitch, D. Revealed: operation shady RAT. White Paper, 2011. `http://www.csri.info/wp-content/uploads/2012/08/wp-operation-shady-rat1.pdf`, accessed June 2018.

Alrawi, O. and Mohaisen, A. Chains of distrust: Towards understanding certificates used for signing malicious applications. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 451–456. International World Wide Web Conferences Steering Committee, 2016.

Annachhatre, C., Austin, T. H., and Stamp, M. Hidden markov models for malware classification. *Journal of Computer Virology and Hacking Techniques*, 11(2):59–73, 2015.

Apel, M., Bockermann, C., and Meier, M. Measuring similarity of malware behavior. In *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, pages 891–898. IEEE, 2009.

Arntz, P. Spigot browser hijackers. Website, February 2017. `https://blog.malwarebytes.com/puppum/2017/02/spigot-browser-hijackers/`, accessed December 2018.

Ashenbrenner, S. Known Bad Software Part I - MacKeeper. Website, May 2018. `https://crashsecurity.com/blog/2018/4/20/mackeeper-is-a-scam`, accessed December 2018.

Bailey, M., Oberheide, J., Andersen, J., Mao, Z. M., Jahanian, F., and Nazario, J. Automated classification and analysis of internet malware. In *International Workshop on Recent Advances in Intrusion Detection (RAID)*, pages 178–197. Springer, 2007.

Barrera, D., Kayacik, H. G., van Oorschot, P. C., and Somayaji, A. A methodology for empirical analysis of permission-based security models and its application to Android. In *Proceedings of the 17th ACM conference on Computer and Communications Security*, pages 73–84. ACM, 2010.

Baumgarten, R. Mach-O Malware Analysis: Combatting Mac OSX/iOS Malware with Data Visualization. In *Proceedings of DefCon 21*. 2013.

Baysa, D., Low, R. M., and Stamp, M. Structural entropy and metamorphic malware. *Journal of Computer Virology and Hacking Techniques*, 9(4):179–192, 2013.

Bestuzhev, D. First step in cross-platform Trojan bankers from Brazil done. Website, March 2018. `https://securelist.com/first-step-in-cross-platform-trojan-bankers-from-brazil-done/74051/`, accessed December 2018.

Breen, K. Darkcomet from defense to offense. In *BSides London Security BSides London*. 2015.

Brillinger, D. R. *et al.* John W. Tukey: his life and professional contributions. *The Annals of Statistics*, 30(6):1535–1575, 2002.

Bureau, P.-M. OSX/Flashback. White Paper, September 2012. `https://www.welivesecurity.com/media_files/white-papers/osx_flashback.pdf`, accessed June 2018.

Cabaj, K., Gawkowski, P., Grochowski, K., Nowikowski, A., and Żórawski, P. The impact of malware evolution on the analysis methods and infrastructure. In *Proceedings of the 14th International Conference on Mining Software Repositories*. PTI, IEEE, 2017.

Calleja, A., Tapiador, J., and Caballero, J. A look into 30 years of malware development from a software metrics perspective. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 325–345. Springer, 2016.

Carbon Black. 2015: The most prolific year in history for OS X malware. Technical Report, 2015. `https://www.documentcloud.org/documents/2459197-bit9-carbon-black-threat-research-report-2015.html`, accessed April 2018.

Cavelty, M. D. Cybersecurity research meets science and technology studies. *Politics and Governance*, 6(2):22–30, 2018.

Chen, P., Desmet, L., and Huygens, C. A study on advanced persistent threats. In *Proceedings of the International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

Cohen, F. Computer Viruses. *Computers & Security*, 6(1):22–35, 1987.

Collins, M. Network Security Through Data Analysis: From Data to Action. O'Reilly Media, Inc., ISBN 9781449357900, 2017.

Dagon, D., Gu, G., Lee, C. P., and Lee, W. A taxonomy of botnet structures. In *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC 2007)*, pages 325–339. IEEE, 2007.

Dai, J., Guha, R. K., and Lee, J. Efficient Virus Detection Using Dynamic Instruction Sequences. *Journal of Computers*, 4(5):405–414, 2009.

Damodaran, A., Di Troia, F., Visaggio, C. A., Austin, T. H., and Stamp, M. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, 2017.

Egele, M., Scholte, T., Kirda, E., and Kruegel, C. A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys (CSUR)*, 44(2):6, 2012.

eSentire. eSentire 2017 Q2 Quarterly Threat Report. Technical Report, 2017. `https://www.esentire.com/assets/resources/2017-Q2-ThreatReport.pdf`, accessed December 2018.

F-Secure. Backdoor:OSX/MacKontrol.A. Website, July 2012a. `https://www.f-secure.com/v-descs/backdoor_osx_mackontrol_a.shtml`, accessed June 2018.

F-Secure. Multi-platform backdoor lurks in colombian transport site. Website, July 2012b. `https://www.f-secure.com/weblog/archives/00002397.html`, accessed June 2018.

Filiol, E. Computer viruses: from theory to applications. Springer Science & Business Media, ISBN 9781280427190, 2006.

FireEye. Poison Ivy: Assessing damage and extracting intelligence. Technical Report, 2013. `https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-poison-ivy.pdf`, accessed February 2018.

Frank, S. The case of the stolen source code. Website, May 2017. `https://panic.com/blog/stolen-source-code/`, accessed December 2018.

Gazet, A. Comparative analysis of various ransomware virii. *Journal of Computer Virology and Hacking Techniques*, 6(1):77–90, 2010.

Geniola, A., Antikainen, M., and Aura, T. A large-scale analysis of download portals and freeware installers. In *Nordic Conference on Secure IT Systems*, pages 209–225. Springer, 2017.

Grégio, A. R. A., Afonso, V. M., Filho, D. S. F., Geus, P. L. d., and Jino, M. Toward a taxonomy of malware behaviors. *The Computer Journal*, 58(10):2758–2777, 2015.

Gu, G., Zhang, J., and Lee, W. Botsniffer: Detecting botnet command and control channels in network traffic. *Computer Science and Engineering Faculty Publications, Wright State University*, 2008.

Gupta, A., Kuppili, P., Akella, A., and Barford, P. An empirical study of malware evolution. In *Proceedings of the First International Communication Systems and Networks and Workshops (COMSNETS 2009)*, pages 1–10. IEEE, 2009.

Hang, H., Bashir, A., Faloutsos, M., Faloutsos, C., and Dumitras, T. "Infect-me-not": A User-centric and Site-centric Study of web-based malware. In *Proceedings of the IFIP Networking Conference and Workshops*, pages 234–242. IEEE, 2016.

Howe, A. E., Ray, I., Roberts, M., Urbanska, M., and Byrne, Z. The psychology of security for the home computer user. In *2012 IEEE Symposium on Security and Privacy*, pages 209–223. IEEE, 2012.

Hsieh, S., Wu, P., and Liu, H. Automatic Classifying of Mac OS X Samples. Technical Report, 2016. `https://documents.trendmicro.com/assets/wp/wp-automatic-classifying-of-mac-os-x-samples.pdf`, accessed April 2018.

Hurier, M., Allix, K., Bissyandé, T. F., Klein, J., and Le Traon, Y. On the lack of consensus in anti-virus decisions: Metrics and insights on building ground truths of android malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 142–162. Springer, 2016.

Hurier, M., Suarez-Tangil, G., Dash, S. K., Bissyandé, T. F., Traon, Y. L., Klein, J., and Cavallaro, L. Euphony: harmonious unification of cacophonous anti-virus vendor labels for android malware. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 425–435. IEEE Press, 2017.

Im, G. P. and Baskerville, R. L. A longitudinal study of information system threat categories: the enduring problem of human error. *ACM SIGMIS Database*, 36(4):68–79, 2005.

Irwin, B. A Source Analysis of the Conficker Outbreak from a Network Telescope. *SAIEE Africa Research Journal*, 104(2):38–53, 2013.

Jackson, K. Nomenclature for malicious programs. *Virus Bulletin*, March 1990.

Jacob, G., Debar, H., and Filiol, E. Behavioral detection of malware: from a survey towards an established taxonomy. *Journal of Computer Virology and Hacking Techniques*, 4(3):251–266, 2008.

Johnston, A. C., Warkentin, M., McBride, M., and Carter, L. Dispositional and situational factors: influences on information security policy violations. *European Journal of Information Systems*, 25(3):231–251, 2016.

Joshi, R. and Pilli, E. S. Botnet forensics. In *Fundamentals of Network Forensics*, pages 145–165. Springer, ISBN 9781447172970, 2016.

Kantchelian, A., Tschantz, M. C., Afroz, S., Miller, B., Shankar, V., Bachwani, R., Joseph, A. D., and Tygar, J. D. Better malware ground truth: Techniques for weighting anti-virus vendor labels. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, pages 45–56. ACM, 2015.

Karim, M. E., Walenstein, A., Lakhotia, A., and Parida, L. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1(1-2):13–23, 2005.

Karresand, M. Separating trojan horses, viruses, and worms-a proposed taxonomy of software weapons. In *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*, pages 127–134. IEEE, 2003.

Kaspersky. Kaspersky security bulletin 2018. Technical Report, December 2018. `https://securelist.com/kaspersky-security-bulletin-2018-statistics/89145/`, accessed December 2018.

Kaspersky, E. The evolution of OS X malware. Website, September 2014. `https://eugene.kaspersky.com/2014/09/29/the-evolution-of-os-x-malware/`, retrieved November 2017.

Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., and Kirda, E. Cutting the gordian knot: A look under the hood of ransomware attacks. In *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer, 2015.

Khattak, S., Ramay, N. R., Khan, K. R., Syed, A. A., and Khayam, S. A. A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys & Tutorials*, 16(2):898–924, 2014.

Kim, D., Kwon, B. J., and Dumitraş, T. Certified Malware: Measuring Breaches of Trust in the Windows Code-Signing PKI. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1435–1448. ACM, 2017.

Kolbitsch, C., Comparetti, P. M., Kruegel, C., Kirda, E., Zhou, X.-y., and Wang, X. Effective and efficient malware detection at the end host. In *Proceedings of the USENIX security symposium*, volume 4, pages 351–366. 2009.

Kolter, J. Z. and Maloof, M. A. Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, 7(Dec):2721–2744, 2006.

Kornblum, J. Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, 3:91–97, 2006.

Kotzias, P., Matic, S., Rivera, R., and Caballero, J. Certified pup: abuse in authenticode code signing. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 465–478. ACM, 2015.

Kwon, B. J., Mondal, J., Jang, J., Bilge, L., and Dumitras, T. The dropper effect: Insights into malware distribution with downloader graph analytics. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1118–1129. ACM, 2015.

Landwehr, C. E., Bull, A. R., McDermott, J. P., and Choi, W. S. A taxonomy of computer program security flaws. *ACM Computing Surveys (CSUR)*, 26(3):211–254, 1994.

Latif, M. H. OSX.Coinminer. Website, February 2018. `https://www.symantec.com/security-center/writeup/2018-021215-4916-99`, accessed June 2018.

Lau, B., Jang, Y., Song, C., Wang, T., Chung, P. H., and Royal, P. Mactans: Injecting malware into ios devices via malicious chargers. *Black Hat USA*, 2013.

Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710. 1966.

Lever, C., Kotzias, P., Balzarotti, D., Caballero, J., and Antonakakis, M. A lustrum of malware network communication: Evolution and insights. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pages 788–804. IEEE, 2017.

Levin, J. Code signing – hashed out. In *Proceedings of the RSA Conference, San Francisco*. 2015.

Levin, J. Mac OS X and iOS Internals, 2nd Edition. Technologygeeks.com, ISBN 099105556X, 2018.

Liu, J., Wang, Y., and Wang, Y. Inferring phylogenetic networks of malware families from api sequences. In *Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 14–17. IEEE, 2016.

Luszcz, J. How maverick developers can create risk in the software and IoT supply chain. *Network Security*, 2017(8):5–7, 2017.

Ma, J., Dunagan, J., Wang, H. J., Savage, S., and Voelker, G. M. Finding diversity in remote code injection exploits. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 53–64. ACM, 2006.

Malin, C. H., Casey, E., and Aquilina, J. M. Malware forensics: investigating and analyzing malicious code. Syngress, ISBN 9780080560199, 2008.

Malwarebytes. Malicious download installs Genieo and GoPhoto.it adware. Website, November 2013. `https://www.thesafemac.com/malicious-download-installs-genieo-and-gophoto-it-adware/`, accessed December 2018.

Malwarebytes. Adware Removal Guide: Bundlore. Website, November 2015. `http://www.thesafemac.com/arg-bundlore/`, accessed June 2018.

Malwarebytes. OSX.Genieo. Website, 2018a. `https://blog.malwarebytes.com/detections/osx-genieo/`, accessed December 2018.

Malwarebytes. PUP.Optional.InstallCore. Website, 2018b. `https://blog.malwarebytes.com/detections/pup-optional-installcore/`, accessed December 2018.

Manning, J. Apple drops claim that "Macs don't get viruses". Website, July 2012. `https://www.smh.com.au/technology/apple-drops-claim-that-macs-dont-get-viruses-20120703-21ei4.html`, accessed October 2018.

Mansfield-Devine, S. Ransomware: the most popular form of attack. *Computer Fraud & Security*, 2017(10):15–20, 2017.

Marquis-Boire, M., Marschalek, M., and Guarnieri, C. Big game hunting: the peculiarities in nation-state malware research. In *Proceedings of Black Hat USA*. 2015.

McAfee. Meet 'Tox': Ransomware for the Rest of Us. Website, May 2015. `https://securingtomorrow.mcafee.com/mcafee-labs/meet-tox-ransomware-for-the-rest-of-us/`, accessed February 2018.

McFedries, P. Technically speaking: The spyware nightmare. *IEEE Spectrum*, 42(8):72–72, 2005.

Mezzour, G., Carley, K. M., and Carley, L. R. Global variation in attack encounters and hosting. In *Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp*, pages 62–73. ACM, 2017.

MITRE. Common malware enumeration. Website, 2006. `http://cme.mitre.org/`, accessed April 2018.

MITRE. Malware Attribute Enumeration and Characterization. Website, 2018. `http://maecproject.github.io//`, accessed April 2018.

Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N. Inside the slammer worm. *IEEE Security & Privacy*, 99(4):33–39, 2003.

Moore, T. The economics of cybersecurity: Principles and policy options. *International Journal of Critical Infrastructure Protection*, 3(3):103–117, 2010.

Moore, T., Clayton, R., and Anderson, R. The economics of online crime. *Journal of Economic Perspectives*, 23(3):3–20, 2009.

Moser, A., Kruegel, C., and Kirda, E. Limits of static analysis for malware detection. In *Proceedings of the 23rd annual Computer Security Applications Conference (ACSAC 2007)*, pages 421–430. IEEE, 2007.

Moubarak, J., Chamoun, M., and Filiol, E. Comparative study of recent MEA malware phylogeny. In *Proceedings of the 2nd International Conference on Computer and Communication Systems (ICCCS)*, pages 16–20. IEEE, 2017.

Nachenberg, C. S. and Griffin, K. E. Reducing malware signature set size through server-side processing. August 7 2012. US Patent 8,239,944.

Oliver, J., Cheng, C., and Chen, Y. TLSH–a locality sensitive hash. In *Proceedings of the Fourth Cybercrime and Trustworthy Computing Workshop*, pages 7–13. IEEE, 2013.

Ollmann, G. The evolution of commercial malware development kits and colour-by-numbers custom malware. *Computer Fraud & Security*, 2008(9):4–7, 2008.

Ormandy, T. Sophail: Applied attacks against Sophos Antivirus. Technical Report, 2011. `https://lock.cmpxchg8b.com/sophail.pdf`, accessed February 2018.

Pagani, F., Dell'Amico, M., and Balzarotti, D. Beyond Precision and Recall: Understanding Uses (and Misuses) of Similarity Hashes in Binary Analysis. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pages 354–365. ACM, 2018.

Pai, S., Di Troia, F., Visaggio, C. A., Austin, T. H., and Stamp, M. Clustering for malware classification. *Journal of Computer Virology and Hacking Techniques*, 13(2):95–107, 2017.

Pajouh, H. H., Dehghantanha, A., Khayami, R., and Choo, K.-K. R. Intelligent OS X malware threat detection with code inspection. *Journal of Computer Virology and Hacking Techniques*, pages 1–11, 2017.

Pham, D. V., Halgamuge, M. N., Syed, A., and Mendis, P. Optimizing windows security features to block malware and hack tools on USB storage devices. In *Proceedings of the Progress in Electromagnetics Research Symposium*, pages 350–355. 2010.

Poston, R. J. Method and system for classification of software using characteristics and combinations of such characteristics. January 2013. US Patent 8,365,286.

Proofpoint. Protecting People: A Quarterly Analysis of Highly Targeted Attacks. Technical Report, July 2018. `https://www.proofpoint.com/sites/default/files/pfpt-uk-tr-people-report-summer-2018-180904.pdf`, accessed December 2018.

Provos, N., McNamee, D., Mavrommatis, P., Wang, K., Modadugu, N. *et al.* The ghost in the browser: Analysis of web-based malware. *Proceedings of the First Workshop on Hot Topics in Understanding Botnets (HotBots '07)*, 7:4–4, 2007.

Raff, E., Zak, R., Cox, R., Sylvester, J., Yacci, P., Ward, R., Tracy, A., McLean, M., and Nicholas, C. An investigation of byte n-gram features for malware classification. *Journal of Computer Virology and Hacking Techniques*, pages 1–20, 2016.

Reed, T. Genieo installer tricks keychain. Website, March 2016a. `https://blog.malwarebytes.com/cybercrime/2015/08/genieo-installer-tricks-keychain/`, accessed December 2018.

Reed, T. PUP Friday: MacKeeper. Website, December 2016b. `https://blog.malwarebytes.com/puppum/2016/08/pup-friday-mackeeper/`, accessed June 2018.

Reed, T. Code signing flaw in macOS. Virus Bulletin Conference Talk, 2018a. `https://papers.put.as/papers/macosx/2018/Codesigning_Mac.pdf`, accessed December 2018.

Reed, T. A data-driven look at the mac threat landscape. MacAdmins Conference at Penn State University, July 2018b. `https://macadmins.psu.edu/files/2018/07/psumac2018-A-data-driven-look-at-the-Mac-threat-landscape-1hauqlq.key`, accessed October 2018.

Reed, T. New Mac cryptominer distributed via a MacUpdate hack. Website, February 2018c. `https://blog.malwarebytes.com/threat-analysis/2018/02/new-mac-cryptominer-distributed-via-a-macupdate-hack/`, accessed December 2018.

Riau, C. A virus by any other name: Virus naming practices. Website, 2002. `https://www.symantec.com/connect/articles/virus-any-other-name-virus-naming-practices`, accessed April 2018.

Rieck, K., Holz, T., Willems, C., Düssel, P., and Laskov, P. Learning and classification of malware behavior. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 108–125. Springer, 2008.

Rodríguez, R. J. Evolution and characterization of point-of-sale ram scraping malware. *Journal of Computer Virology and Hacking Techniques*, 13(3):179–192, 2017.

Rosenberg, I., Sicard, G., and David, E. O. Deepapt: nation-state apt attribution using end-to-end deep neural networks. In *International Conference on Artificial Neural Networks*, pages 91–99. Springer, 2017.

Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H., and Van Steen, M. Prudent practices for designing malware experiments: Status quo and outlook. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pages 65–79. IEEE, 2012.

Roussev, V. Data fingerprinting with similarity digests. In *Proceedings of the IFIP International Conference on Digital Forensics*, pages 207–226. Springer, 2010.

Scheidl, G. Virus Naming Convention 1999 (VNC99). Website, July 1999. `http://members.chello.at/erikajo/vnc99b2.txt`, accessed April 2018.

Sebastián, M., Rivera, R., Kotzias, P., and Caballero, J. Avclass: A tool for massive malware labeling. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*, pages 230–253. Springer, 2016.

Seideman, J. D., Khan, B., and Vargas, A. C. Malware biodiversity using static analysis. In **Doss, R., Piramuthu, S., and ZHOU, W.**, editors, *Future Network Systems and Security*, pages 139–155. Springer International Publishing, Cham, 2015a. ISBN 978-3-319-19210-9.

Seideman, J. D., Khan, B., and Vargas, C. Quantifying malware evolution through archaeology. *Journal of Information Security*, 6(02):101, 2015b.

Serper, A. OSX.Pirrit: The Minds Behind the Malicious Mac Adware. Technical Report, June 2016. `http://go.cybereason.com/rs/996-YZT-709/images/Cybereason-Labs-Research-OSX.Pirrit-Minds-Behind-Malicious-Mac-Adware.pdf`, accessed December 2018.

Serper, A. How We Reverse Engineered OSX/Pirrit, Got Legal Threats and Survived. BSidesCharm Presentation, May 2018.

Shamsi, J. A., Zeadally, S., Sheikh, F., and Flowers, A. Attribution in cyberspace: techniques and legal implications. *Security and Communication Networks*, 9(15):2886–2900, 2016.

Shannon, C. E. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

Sikorski, M. and Honig, A. Practical malware analysis: the hands-on guide to dissecting malicious software. No Starch Press, ISBN 9781593272906, 2012.

Simmons, C., Ellis, C., Shiva, S., Dasgupta, D., and Wu, Q. AVOIDIT: A cyber attack taxonomy. In *Proceedings of the 9th Annual Symposium On Information Assurance (ASIA)*, volume 14. 2009.

Skulason, F., Solomon, A., and Bontchev, V. A new virus naming convention. Website, 1991. `http://www.caro.org/articles/naming.html`, accessed April 2018.

Soh, B., Dillon, T. S., and County, P. Quantitative risk assessment of computer virus attacks on computer networks. *Computer Networks and ISDN systems*, 27(10):1447–1456, 1995.

Soto-Valero, C. and González, M. Empirical study of malware diversity in major android markets. *Journal of Cyber Security Technology*, pages 1–24, 2018.

Spafford, E. H. Computer viruses–a form of artificial life? Technical report, Department of Computer Science, Purdue University, 1990.

Spafford, E. H. Computer viruses as artificial life. *Artificial life*, 1(3):249–265, 1994.

Spafford, E. H., Heaphy, K. A., and Ferbrache, D. J. A computer virus primer. In *Computers under attack: intruders, worms, and viruses*, pages 316–355. ACM, ISBN 0201530678, 1991.

Stack Overflow. Developer survey results. Website, 2016. `https://insights.stackoverflow.com/survey/2016`, accessed November 2017.

Stalmans, E. and Irwin, B. A framework for DNS based detection and mitigation of malware infections on a network. In *Proceedings of Information Security South Africa (ISSA)*, pages 1–8. IEEE, 2011.

StatCounter. Desktop operating system market share worldwide. Website, October 2017. `http://gs.statcounter.com/os-market-share/desktop/worldwide#monthly-201704-201704-map`, accessed October 2017.

Stevens, K. and Jackson, D. Zeus banking trojan report. Technical report, SecureWorks Counter Threat Unit, 2010.

Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P., and Ribagorda, A. Evolution, detection and analysis of malware for smart devices. *IEEE Communications Surveys & Tutorials*, 16(2):961–987, 2014.

Symantec. Trojan.emotet. Website, July 2017. `https://www.symantec.com/security-center/writeup/2017-071312-0253-99`, accessed December 2018.

Symantec. 2018 Internet Security Threat Report. Technical Report, March 2018a. `https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf`, accessed December 2018.

Symantec. OSX.Getshell. Website, 2018b. `https://www.symantec.com/security-center/writeup/2013-020412-3611-99`, accessed December 2018.

Szappanos, G. Akbuilder - the crowdsourced exploit kit. Technical report, Sophos, 2016.

Szor, P. The art of computer virus research and defense. Symantec Press, ISBN 9780321623980, 2005.

Tahir, R., Huzaifa, M., Das, A., Ahmad, M., Gunter, C., Zaffar, F., Caesar, M., and Borisov, N. Mining on someone else's dime: Mitigating covert mining operations in clouds and enterprises. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*, pages 287–310. Springer, 2017.

Thonnard, O. and Dacier, M. A framework for attack patterns' discovery in honeynet data. *Digital Investigation*, 5:128–139, 2008.

Tridgell, A. Efficient algorithms for sorting and synchronization. Ph.D. thesis, Australian National University Canberra, 1999.

Tukey, J. W. Exploratory data analysis, volume 2. Pearson, ISBN 0201076160, 1977.

UK Foreign & Commonwealth Office. Foreign Office Minister condemns Russia for NotPetya attacks. Foreign Office Website, February 2018. `https://www.gov.uk/government/news/foreign-office-minister-condemns-russia-for-notpetya-attacks`, accessed February 2018.

Urban, T., Tatang, D., Holz, T., and Pohlmann, N. Towards understanding privacy implications of adware and potentially unwanted programs. In *European Symposium on Research in Computer Security*, pages 449–469. Springer, 2018.

US-CERT. Alert (TA17-132A): Indicators Associated With WannaCry Ransomware. Website, May 2017. `https://www.us-cert.gov/ncas/alerts/TA17-132A`, accessed December 2018.

Van Mieghem, V. Behavioural Detection and Prevention of Malware on OS X. *Virus Bulletin*, 2016.

Velleman, P. F. and Hoaglin, D. C. Applications, basics, and computing of exploratory data analysis. Duxbury Press, ISBN 087150409X, 1981.

Venkatesan, D. Mobile malware factories: Android apps for creating ransomware. Website, August 2017. `https://www.symantec.com/connect/blogs/mobile-malware-factories-android-apps-creating-ransomware`, accessed February 2018.

Vermeulen, J. An analysis of fusing advanced malware email protection logs, malware intelligence and active directory attributes as an instrument for threat intelligence. Master's thesis, Rhodes University, 2018.

Wallace, B. Optimizing ssdeep for use at scale. *Virus Bulletin*, 2015.

Wang, A., Liang, R., Liu, X., Zhang, Y., Chen, K., and Li, J. An inside look at iot malware. In *Proceedings of the International Conference on Industrial IoT Technologies and Applications*, pages 176–186. Springer, 2017.

Wardle, P. Methods of malware persistence on Mac OS X. In *Proceedings of the Virus Bulletin Conference*. 2014.

Wardle, P. Offensive Malware Analysis: Dissecting OS X FruitFly. In *Proceedings of DefCon 25*. 2017a.

Wardle, P. OSX/MacRansom. Website, 2017b. `https://objective-see.com/blog/blog_0x1E.html`, accessed December 2018.

Wardle, P. Mac Malware. Website, December 2018a. `https://objective-see.com/malware.html`, accessed December 2018.

Wardle, P. Protecting the Garden of Eden. In *Proceedings of the "Objective by the Sea" macOS Security Conference*. 2018b.

Wardle, P. Word to your mac. Video recorded session of a malicious document analysis, December 2018c. `https://youtu.be/UD2-rc2itF0`, accessed December 2018.

Wash, R. Folk models of home computer security. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, page 11. ACM, 2010.

Weaver, N., Paxson, V., Staniford, S., and Cunningham, R. A taxonomy of computer worms. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, pages 11–18. ACM, 2003.

Wong, W. and Stamp, M. Hunting for metamorphic engines. *Journal in Computer Virology*, 2(3):211–229, 2006.

Xiao, C. and Chen, J. New OS X ransomware KeRanger infected Transmission BitTorrent client installer. Website, March 2016. `https://unit42.paloaltonetworks.com/new-os-x-ransomware-keranger-infected-transmission-bittorrent-client-installer/`, accessed June 2018.

Ye, Y., Li, T., Adjeroh, D., and Iyengar, S. S. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)*, 50(3):41, 2017.

Yen, T.-F., Heorhiadi, V., Oprea, A., Reiter, M. K., and Juels, A. An epidemiological study of malware encounters in a large enterprise. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1117–1130. ACM, 2014.

Yonts, J. Mac OS X Malware Analysis. SANS Technical Report, 2009. `https://www.sans.org/reading-room/whitepapers/forensics/mac-os-malware-analysis-33178`, accessed April 2018.

You, I. and Yim, K. Malware obfuscation techniques: A brief survey. In *Proceedings of the International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA)*, pages 297–300. IEEE, 2010.

Zanero, S. Behavioral intrusion detection. In *Proceedings of the International Symposium on Computer and Information Sciences*, pages 657–666. Springer, 2004.

Zdziarski, J. Identifying back doors, attack points, and surveillance mechanisms in ios devices. *Digital Investigation*, 11(1):3–19, 2014.

Zhou, Y. and Jiang, X. Dissecting android malware: Characterization and evolution. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 95–109. IEEE, 2012.

Appendix

Source code used in this research can be found at `https://thesis.yaxs.net`. Where applicable, specific files have been linked in footnotes.