Digital Investigation and the Trojan Defense, Revisited

Golden G. Richard III

Center for Computation and Technology and Division of Computer Science and Engineering, Louisiana State University

Andrew Case

Volatility Foundation

Modhuparna Manna

Division of Computer Science and Engineering, Louisiana State University

Elsa A. M. Hahne

Office of Research & Economic Development, Louisiana State University

Aisha Ali-Gombe

Department of Computer and Information Sciences, Towson University

Abstract

Over the past few decades, rapid changes in technology have driven a significant increase in the amount and types of data stored on and processed by digital devices. Digital devices may be used in the commission of numerous criminal activities, including unauthorized data exfiltration, fraud, employee misconduct, kidnapping, child pornography, murder, and more. After being accused of committing a crime, a common defense is the so-called Trojan horse defense. In the Trojan defense, the defendant claims that someone or something else is responsible for the crime committed as represented by digital evidence present on one or more devices. Traditionally, the Trojan defense has often been dismissed by investigators after a cursory examination of digital devices for the presence of malware. While this process might have led to fair conclusions in the past, we now face increasingly sophisticated cyber attacks and malware infections, and it is increasingly possible that someone or something (e.g., malware) other than the "obvious" party may be guilty. This chapter discusses the impact of modern malware on digital investigations and examines possible solutions to the problem of unraveling the accuracy of the Trojan defense, including the use of memory forensics techniques.

Keywords: memory forensics, computer forensics, Trojan defense, digital investigation, malware

1. Introduction

In recent times, digital forensics capabilities have been significantly expanded through the development of a number of new tools and techniques. Modern forensic tools can reveal data in plain sight (e.g., files containing copies of credit card statements, spreadsheets), data that was previously deleted by users (files, SMS messages, logs, etc.), illicit data (NSFW materials, sensitive documents which the user is not authorized to possess, digital contraband, etc.), evidence that systems were used to attack others (e.g., command histories), geo-location information, and more. Increasingly, digital forensics tool suites support "pushbutton forensics," which allow for rapid recovery of digital evidence, data correlation, creation of timelines, and selective acquisition of evidence without significant effort, or in some cases, without significant expertise on the part of investigators. Making digital forensics tools easier to use and automating tedious investigatory processes is undoubtedly useful, as it reduces investigator fatigue and case backlogs. But there is also a significant downside. As digital forensics techniques have evolved, so has the design and capabilities of modern malware. It is becoming increasingly difficult to conduct digital investigations correctly, and in the face of sophisticated malware, traditional storage forensics methods are no longer sufficient to refute the Trojan defense.

Historically, malware caused disruption primarily by deleting data and limiting the performance or capabilities of computing devices (Rankin, 2018). Furthermore, the incentives behind development and deployment of historical malware were often unclear. In sharp contrast, the design and development of modern malware is usually motivated by a number of distinct factors, including the potential for monetary gain or commercial advantage, revenge, the needs of nation-state actors, and more. To this end, modern malware frequently alters the state of computing devices, infiltrates

Email addresses: golden@cct.lsu.edu (Golden G. Richard III), andrew@dfir.org (Andrew Case), mmanna3@lsu.edu (Modhuparna Manna), ehahne@lsu.edu (Elsa A. M. Hahne), aaligombe@towson.edu (Aisha Ali-Gombe)

and exfiltrates data, and performs unauthorized activities "on behalf of" users, such as web surfing, sending email, and downloading files. Detection of modern malware is neither straightforward nor certain, particularly if only traditional digital forensics techniques are utilized. These techniques typically examine only the contents of non-volatile storage, whereas many strains of modern malware and attack toolkits leave absolutely no traces on disk (Kaspersky Research Team, 2014, 2015; Schroeder, 2019; Wadner, 2014). Thus, the impact modern malware can have on innocent users is enormous, as malware can perform virtually any action that a user might perform, including the download of illicit or illegal materials, such as child pornography, without being easily detected. Furthermore, while personal security products such as antivirus programs are adept at detecting historical, well-known, and established malware, detection rates for new and emergent strains remain notoriously low. Thus, "personal computer hygiene" is insufficient as a defensive measure against modern malware.

For individuals accused of wrongdoing involving digital devices, there is a very substantial burden in defending themselves when expensive technical expertise is required to recover exculpatory evidence such as proof of a malware infection. We argue that not only is a deliberate and earnest search for malware a necessary component of most digital forensics cases (where we strongly agree with (Bowles and Hernandez-Castro, 2015)), but traditional forensics techniques *must* be supplemented with modern analysis techniques to "balance the scales." The most appropriate and powerful of modern techniques is memory forensics, which deeply examines the state of a system through analysis of volatile memory (RAM). By leveraging memory forensics, investigators can uncover a wealth of information that is not recorded in the file system, including signs of malware infection. Such analysis performed by experts is necessary to remove the burden of proof from suspects who have no realistic chance of discovering, analyzing, or documenting the malware capabilities themselves. This is a complicated issue, however, as substantial time and expertise is needed to properly conduct thorough memory forensics investigations across a variety of devices, operating system versions, and applications. As such, the resources and costs associated with this kind of analysis can quickly become daunting and prohibitive.

The rest of the chapter is organized as follows: Section 2 discusses the Trojan defense in more detail. Section 3 briefly surveys traditional digital forensics methodologies. Section 4 traces the evolution of modern malware and the challenges posed for digital forensics investigation, including disentangling user actions and malware effects. Section 5 outlines memory forensics capabilities and how memory forensics can be used to more accurately detect malware. Section 6 provides some concluding thoughts.

2. The Trojan Defense

Cybercrimes have escalated significantly over the past two decades (Cybercrime, 2019). The entire infrastructure of computer systems and networks has changed dramatically, and crimes that once were somewhat simple to detect are now much more complicated. Law enforcement faces serious challenges, on multiple levels, in combating such crimes. While the world becomes closer and closer connected through the internet and criminals routinely access their victims' computers remotely and escape without leaving any identifiable trace, law enforcement must operate in accordance with local, state, or federal cyberlaws, which vary from one geographic area to the next (Luehr, 2005). Many defendants charged with cybercrimes cite the "Trojan horse defense," stating that, without their knowledge, a malicious cybercriminal hacked into their computer to commit the crime, or planted malware responsible for said crime (Brenner et al., 2004).

Suppose someone named David is charged with possession of child pornography on his computer. If David pleads the Trojan defense, his claim is that he did not intentionally or knowingly download this content. Instead, an anonymous and unidentifiable cybercriminal or malware placed the material on his computer. Because it is often difficult or impossible for prosecutors to refute this defense, that is, proving the defendant is responsible for downloading the illegal content, many of them escape either prosecution or conviction. For example, Aaron Caffrey was charged with unauthorized computer modifications for launching a DDoS (Distributed Denial of Service) attack against a fellow chatroom user, "Bokkie." The DDoS attack passed through several servers and brought traffic at the Port of Houston to a dangerous standstill. Caffrey, however, argued that while the attack was launched from his computer, he himself did not launch it; instead, it was launched by a hacker group that had surreptitiously planted trojan malware on his computer and later wiped it away. Even though Caffrey had no evidence to substantiate his theory, he was still acquitted simply because the prosecution had no evidence to rebut it (Brenner et al., 2004).

Although the Trojan defense results in acquittals in some cases, it is not always successful. There are examples of cases where the defendant claimed the Trojan defense and still got convicted. In the Mark Rawlinson case, for instance, the defense claimed a virus was responsible for downloading the thousand pornographic photos found on his computer. The court did not accept the Trojan defense in this case, and Rawlinson was convicted (Bowles and Hernandez-Castro, 2015).

There is an ongoing disagreement among academics, practitioners, and law enforcement about whether malware attacks can result in an innocent person being ultimately held responsible for crimes like possession of child pornography, tax fraud, DDoS, and other attacks. For instance, in the Miller case. When Miller was charged with possession of child pornography, he contacted the FBI agent in charge claiming a virus was responsible for the crime. Agent Kyle, the investigator in charge, refuted the Trojan defense arguments saying a virus could not do such a thing (Monterosso, 2010). This might be true in some clear-cut cases, however, the modern scenario has completely changed. Nowadays, malware has become sophisticated enough to download child pornography, modify the browser information stored in the temporary cache locations, send out unwanted emails, and so on. Consider the Matthew Bandy case, where Bandy was originally charged with life imprisonment for possession of child porn until forensic investigator Tami Loehrs found more than 200 malware-infected files in his computer. After some negotiation and with the help of Loehrs findings, Bandy was finally given an 18-month probation period, during which he had to register as a sex-offender (Mcelroy, 2007). The Trojan defense, therefore, makes cybercrime cases highly complicated.

We emphasize that even if the defendant is finally acquitted, it does not undo much of the damage already done. Just being charged with a crime like possession of child pornography can be very costly, from both monetary and psychological points of view. The harm can include severe emotional stress, loss of reputation, employment, financial resources, and emotional support. When Michael Fiola was charged with child pornography, he lost his job with the Massachusetts Department of Industrial Accidents and was abandoned by both family and friends. Even though Fiola was ultimately acquitted, he never got his job back, and many of his relationships remain strained (PCWorld, 2008).

Many authors have discussed cases where the Trojan defense played a prominent role, outlining outcomes, various legal loopholes, and how justice can somehow be guaranteed in spite of the added complications associated with this defensive strategy. One of the seminal reports in this area was drafted by Susan W. Brenner and her colleagues (Brenner et al., 2004). The report discusses the Trojan horse defense cases of Aaron Caffrey, Julian Green, Karl Schofield, and Eugene Pitts in detail, and highlights the steps the prosecution could take to verify if the defendant is guilty. This report is almost invariably cited whenever the Trojan horse defense is discussed. According to the authors, in case of a Trojan defense, there are several possible ways that the prosecution could proceed. First, the prosecution might insist that since "malware did it," the defense should produce evidence that malware is present and was installed by someone else without the defendant's knowledge. For reasons we will discuss later in the paper, our position is that while this may have been possible in the past, modern malware can be extremely difficult to detect; detection likely requires substantial technical skill and the use of forensic procedures not commonly used by law enforcement. Furthermore, transient, memory-only malware may leave no trace at all, making the production of evidence of the malware's existence virtually impossible. Another tactic that is proposed for use by the prosecution is to establish the level of computer expertise of the defendant and find out if they possessed enough technical expertise to avoid becoming a victim of malware. As we will discuss later in this chapter, the bar for self-protection is spectacularly high when sophisticated modern malware is concerned. Finally, the assertion in the Brenner paper that law enforcement can "negate the factual basis of the defense" by establishing that no malware is present must go far beyond the use of traditional forensics techniques and antivirus scans.

Though there are many theories as to how prosecutors could deal with the defendants in a Trojan defense case, things may not be so simple in practice. Each case is different, and there are no straight-forward, general recommendations for how to handle any particular case. Furthermore, forensic investigations can go terribly wrong, allowing malware to go undetected. In Julie Amero's case, she was charged with showing indecent materials to twelve-year-old kids during a class. While she was teaching, her computer started showing indecent pictures downloaded by NewDotNet spyware. Amero panicked and went to seek help while the pictures kept showing. Though the spyware was responsible for the crime, it took years for Amero to get justice. Part of the reason for her delayed justice is that the digital forensic investigation was not carried out properly. Her hard drive was imaged using the backup utility Ghost, which is not considered standard for disk imaging (Eckelberry et al., 2007). Although Ghost can be configured to perform sector-level copies of hard drives, default options must be overridden to avoid capturing only logical volume copies in at least some versions. In cases of negligence on the part of the investigator, or in the presence of highly sophisticated malware, investigations could easily fall short. Therefore, the presence of computer forensic experts as well as appropriate tools and techniques for forensic investigation of Trojan horse defense cases are extremely valuable.

Paul H. Luehr elaborates on where and how forensic investigators can find relevant evidence in cases involving digital systems. Luehr points out places such as start-up configuration files, internet browsing histories, and other places where evidence impacting a Trojan defense case might be found (Luehr, 2005). Haagman and Ghavalas mention the importance of volatile and network evidence in cases of cybercrimes involving a Trojan defense (Haagman and Ghavalas, 2005). The second part of the paper describes how these techniques can be used to analyze the presence of a backdoor (Ghavalas and Philips, 2005).

In the past 10 years, the effectiveness of the Trojan defense has varied significantly. It has not been particularly successful in United States criminal cases, with no published acquittals in cases where it was the primary defense. On the civil side, the defense has met with more success (Steel, 2014). Bowles and Hernandez-Castro have published several case studies where the defendant was able to prove that malware was present without their knowledge (Bowles and Hernandez-Castro, 2015). No matter the outcome, however, thorough forensic investigations involving sophisticated tools and techniques were required.

3. Traditional Digital Forensics Investigation

Digital forensics is the branch of forensic sciences that deals with the acquisition, analysis, extraction, preservation, and presentation of digital evidence (Carrier, 2003). The field is defined by sets of procedures, tools, and techniques for preserving and analyzing digital evidence recovered from a wide variety of digital devices. While the associated legal requirements governing issues such as right-to-investigate, chain of custody, qualifications for expert witnesses, and admissibility of evidence are important, we focus solely on some relevant technical issues associated with acquisition and analysis in this section.

3.1. Traditional Acquisition

In traditional digital forensics, acquisition and recovery of data specifically targets non-volatile storage, such as hard drives. The primary objective is to extract copies of storage media where the copies are as close to "bit-perfect" as possible. For fully functional media in computer systems or devices that have been powered down, bit-perfect copies can be generated fairly easily using one of a variety of methods, such as the use of open-source utilities like `dd`, commercial software like FTK Imager (AccessData, 2019), and hardware media duplicators. The use of write-blocking hardware is generally recommended to avoid accidentally corrupting evidence sources during the acquisition phase. This type of acquisition is commonly known as "full disk forensics," wherein entire copies of hard drives are created in a digital forensics lab, or sometimes, at a crime scene. As new storage devices, such as thumb drives, external hard drives, and SD cards, have become widely used, the same approach to creating exact copies of entire storage media has prevailed using these traditional techniques. Typically, both the original media and the copies of the media to be analyzed are run through a cryptographic hashing process (traditionally, MD5 or SHA-1, but now, more likely a newer standard like SHA-256 or SHA-3) that yields an associated digital fingerprint. This fingerprint can be used to establish that the media and copies remain unaltered by subsequent investigative processes.

A standard procedure in performing acquisition of this type is the use of "dead box forensics." Upon entering a potential crime scene, all of the digital devices are powered off to prevent further changes to file systems and storage media. Depending on the type of device, either the entire device is subsequently transported to a forensics lab, or in some extenuating circumstances, just the non-volatile storage media. This "power off and then copy" approach necessarily destroys volatile evidence that is present only in RAM. The volatile evidence that is lost may include memory-resident malware, unsaved documents, and much more. It is very reasonable to expect that some of this evidence may be exculpatory. We emphasize that this volatile evidence is precisely the domain of memory forensics, which is covered in detail in Section 5.

Beyond the loss of volatile evidence, there are several technical hurdles that must be overcome during traditional acquisition. The first is how imaging of malfunctioning media might be carried out. Hard drives and other media that have been damaged due to physical abuse or normal wear and tear frequently exhibit errors during the copying process. Depending on the severity of the errors, various remedies may be required, from simply ignoring "bad" regions of the media and substituting zeros in the copy to attempts to physically repair the media, or use of sophisticated "deep" imaging provided by devices such as Deepspar's PC-3000 (DEEPSPAR Data Recovery Systems, 2019), which attempt to maximize data recoverable from damaged drives by reading sectors in different orders, powering the drive up and down, and more. In many cases, these repairs are very effective; with minimal effort, storage devices can be revived. The authors have direct experience with cases in which opposing counsel in a civil case asserted that damaged media was completely unusable, but with simple replacement of electronic components, all of the stored data could be recovered. Importantly, the data on the damaged storage device had a crucial impact on the outcome of the case.

Nowadays, an even more serious hurdle commonly encountered in the acquisition phase of an investigation is related to the increasing size of modern storage media, which long ago passed a critical point. When hard drives and other storage media were significantly smaller (e.g., no larger than 10s of gigabytes), full acquisition was both feasible and relatively time-efficient. With single commodity hard drives now exceeding 15TB, inexpensive network-attached storage devices allowing users to easily create drive arrays with 100s of TBs of storage, and individual drives up to 100TB on the horizon, making bit-perfect, full copies of media can be extremely time-consuming and in many cases infeasible. Acquisition strategies that copy only "relevant" portions of storage media in lieu of complete copies have been proposed (Grier and Richard, 2015), but particular care must be taken if these selective approaches are used to generate primary copies of storage devices, or critical exculpatory evidence could easily be missed.

3.2. Traditional Analysis

Once a forensically sound copy of a storage medium has been made, forensic analysis can begin. A typical digital forensics investigation involves a number of steps, most of which are generally carried out in a commercial forensics suite, such as those offered by AccessData (AccessData, 2019), Blackbag (Blackbag Technologies, 2019), or OpenText Security (OpenText Security, 2019), potentially supplemented with additional standalone tools to handle data like email or databases. The investigative steps include indexing of the storage media under investigation to allow for fast keyword searches of documents and unallocated space; file carving to recover deleted files (Richard and Roussev, 2005); timelining to determine when specific files were accessed or modified; scrutiny of the Windows registry (when computers running Microsoft Windows are being investigated) for evidence that external storage devices were recently used and which documents were recently accessed (Carvey, 2019); investigation of web browsing history; and more. These forensics procedures are quite adept at revealing incriminating evidence (e.g., downloaded child porn or other illicit materials, web surfing activity) without necessarily establishing that a user performed these actions.

Of particular importance is the investigation of persistence mechanisms used by malware, such as modification of the Windows registry *Run* keys. Various *Run* keys specify the applications that run each time Windows is booted, for each user and for all users. Since this facility is well-known, malware that persists using this technique frequently uses an application name that closely matches commonly installed benign applications. Fortunately, scrutiny of the *Run* keys at the very least reveals applications that are deserving of further investigation. The authors have personal experience with a case in which an employee that was the victim of targeted malware was ultimately *not* terminated based solely on the discovery of the malware via its use of Windows registry *Run* keys for persistence and a subsequent complex reverse engineering effort. All of the other evidence pointed squarely at the employee accessing NSFW materials in the workplace, which was grounds for termination. Importantly, the malware sample that was discovered was *not* detected by antivirus. Screening of the sample using the Virus Total website revealed that virtually all antivirus products flagged the sample as benign, while the few that marked it as suspicious inaccurately described its behavior (e.g., as a banking trojan). Only a detailed reverse engineering effort revealed that the malware surreptitiously accessed pornographic websites "on behalf of" the innocent employee. While a full treatment of persistence mechanisms is beyond the scope of this chapter, a good treatment for Windows can be found in (Fortuna, 2017).

Unfortunately, a huge number of persistence mechanisms exist. Most are neither as obvious nor as well-documented as the use of *Run* keys. Worse, modern memory-only malware may not use any persistence mechanisms at all. Our contention is that while traditional digital forensics techniques are both useful and essential, they miss a significant portion of the narrative surrounding a potential violation of law or policy involving digital devices.

4. Modern Malware and the Trojan Defense

Malware is software that performs unwanted or illicit activities on digital devices. Both the installation of malware and the actions performed by malware are generally without user consent. Over the years, malware has evolved from simple viruses designed as "pranks" to drivers for launching targeted and specialized attacks against individuals, corporations, and cyberinfrastructure. Modern malware is used on a massive scale by cybercriminals using sophisticated capabilities that can target user and corporate data, industrial control systems, equipment used for national defense, and more. These new malware attacks are explicitly designed to violate the security of a target system by breaching confidentiality, violating data and system integrity, and hiding their actions. The actions performed by modern malware include virtually anything a user might do, as well as sophisticated data infiltration and exfiltration, and attacks against other computer systems.

Broadly, malware is categorized by its mode of propagation; specifically, as viruses, worms, and trojans. We explore these briefly, although specific malware samples often exhibit behavior that spans more than one category:

- *Viruses* are malicious software that are self-propagating but not self-contained, meaning they require a host program to exist. However, they can move from one system to another by infecting programs that are subsequently copied to another machine. Viruses are the oldest form of malware that are specifically designed to inject themselves into an existing program by modifying the target binary to include the malicious code.

- *Worms* are malware that are both self-contained and self-propagating. This means that worms exist as independent pieces of code that do not require a host to exist. Usually, worms propagate to a target machine over a network.

- *Trojans* typically possess both an overt and covert functionality. Often propagated via a drive-by download or via social engineering tricks, this category of malware might be installed on user systems as legitimate applications, but unbeknownst to the user, they also have built-in malicious functionality.

Malware can also be classified by their payload or malicious behaviors. Some of these classifications include:

- *Ransomware* encrypts user files in exchange for ransom, which is typically paid using cryptocurrency.

- *Keystroke loggers* surreptitiously capture what a user types at the keyboard, including login credentials and other private data.

- *Spyware* logs and exfiltrates information about user activities, video, audio, etc.

- *Botnets* are large groups of infected machines that can be used for distributed attacks, such as mass spam campaigns and denial of service attacks.

- *Rootkits* modify the system configuration and potentially the operating system to hide its presence and ensure that attackers have continued access to a system.

According to PandaLabs, an average of 230,000 new malware samples were produced daily in 2015, and AV-TEST Institute has reported more than a 20-fold increase in registered malware from 2010 to 2019 (AV-Test, 2019). The development and dissemination of malicious software has become a lucrative, international business with cybercriminals providing malware-as-a-service via the leasing of both software and hardware needed to execute a cyberattack (Laing, 2018). A number of mechanisms are used to install malware on user systems, including email attachments, social engineering, drive-by downloads, vulnerable websites, and more. Modern malware is notorious for installing backdoor programs, spying on user behavior, performing ransomware attacks, mining cryptocurrency, downloading illicit materials, and more. As one example, the Emotet Trojan leverages vulnerable web services to steal financial data, inject a victim's machine with an exploit, and download additional malware to serve as spyware and/or as a backdoor (Malwarebytes Labs, 2019), all without the user's knowledge. On mobile devices, malware has become a critical problem. According to the annual Internet Security Threat Report (ISTR) in 2018, the number of new mobile malware variants has increased by 54% (Symantec, 2018). Android malware such as Cerberus (Doffman, 2019) leverages fake websites and masquerades as a legitimate Adobe Flash Player installation. Upon installation, the malware tricks the user into granting very dangerous permissions, which then gives the attacker access to the device's screen when other legitimate apps are running. With this capability, the malware can steal user keystrokes and sensitive data, such as login credentials and contact information.

4.1. Malware Obfuscation

Anti-virus software is often used to detect malware and prevent new infections. This typically relies on signature-based detection of known malware variants. While anti-virus software is quite effective at detecting these, it frequently fails to detect new ones, particularly targeted malware and those that exhibit advanced obfuscation techniques, such as polymorphism, metamorphism, code and data encryption, kernel manipulation, and most worrisome, memory-only activity. In the case of a new malware, anti-virus software employs heuristic-based approaches to determine known malicious traits (Microsoft, 2011).

Before malware samples can be analyzed, the malware must first be detected and an appropriate sample isolated for further scrutiny. This is the single most pressing issue in validating or invalidating a Trojan defense claim in a case where modern malware could be involved.

On traditional computers, advanced malware, such as kernel-level rootkits, can evade most forms of detection used in traditional malware forensics investigations. Given their power over a system, rootkits can directly choose what data is written to disk and how it is written to disk. Rootkits leverage the power to write as little of its related data to disk as possible and usually choose to encrypt or obfuscate this data. Another category of highly obfuscated malware are fileless malware. These do not create new executable files on disk, but instead embed malicious scripts inside existing files, such as shortcut files or within the registry (Armerding, 2013; Majumder, 2019). Detection of fileless malware requires new detection algorithms as there are no traditional executables to scan and analyze. Instead, anti-virus and other security products must implement completely new features to scan the new sources of malicious code. A recent and prominent example of a fileless malware attack was the hack on the Democratic National Convention (DNC), which leveraged a Powershell backdoor with a persistence mechanism injected into the Windows Management Instrumentation (Alperovitch, 2016). For traditional computers, the most dangerous form of malware is memory-only malware, which does not persist past reboot nor write any data to non-volatile storage. Detection of such malware requires the use of memory forensics since traditional forensics techniques will completely miss all related artifacts. Duqu2 (Kaspersky Research Team, 2015) is one of the most famous malware samples to employ these techniques.

On mobile platforms, other specialized programming practices such as Java reflection and the use of dynamic class loading are becoming standard techniques for mobile malware obfuscation. These techniques can help malware thwart analysis and obfuscate its payload at installation. One of the latest Android malware samples, called "Joker," was detected in September 2019 on the GooglePlay store after more than 400,000 downloads (O'Donnell, 2019). It has both Trojan and spyware capabilities and employs dynamic class loading to install an extra component with more enhanced features capable of subscribing to premium SMS, information stealing, etc.

Given these advanced obfuscation and hiding mechanisms, it is apparent that traditional on-disk malware forensics is under threat as the primary technique for postmortem investigations of cybercrime, especially when stealthy malicious software is involved. We briefly discuss malware analysis in the next section. Evaluating a Trojan defense claim requires *accurately* determining what actions the malware can perform. Without better detection strategies, incorrect decisions regarding Trojan defense claims may be inevitable.

4.2. Malware Analysis

As malware becomes more sophisticated, there is an increasing need to understand not only the malware's actions at the time of execution, but its provenance, persistence, dynamic injection methods, and both internal and external remote communications. Thus, to understand the causality as well as the functionality of a cyberattack in its entirety, especially those involving malware, the security community has adopted a postmortem investigative process called *malware forensics*. This technique involves examining malware code and its effects to identify how an attack happened, what data and resources were compromised, and what actions were executed (Malin et al., 2008). When malware is identified in a case, it is imperative to understand precisely what actions can be attributed to the malware vs. a human user.

Once a malware sample is available, there are two primary approaches: static analysis and dynamic analysis, although these are frequently used together for additional insight. With static analysis, a malware sample's code is examined for any malicious functionality, often using predetermined signatures, patterns, code sequences, semantics, and strings. Static analysis can often be a fast and efficient technique, particularly to understand variants of known malware. Unfortunately, deep static analysis often requires specialized reverse engineering skills and extensive experience, which are in short supply.

Dynamic analysis on the other hand, involves executing a sample in a contained environment and then monitoring its behavior at runtime. With this technique, unwanted or illicit behavior can often be detected and understood much more quickly than via an in-depth static analysis effort. Expert knowledge is typically required in building the execution and analysis environment, although some systems like Cuckoo (Cuckoo Foundation, 2019) ease this burden. While dynamic analysis methodologies such as taint-analysis (Schwartz et al., 2010) perform an excellent job in program tracing, drawbacks include significant resource overhead, contamination of the analysis environment with analyst code, and limited path exploration. Specifically, dynamic analysis may not reveal "hidden" malware behaviors, which are triggered only by certain user actions or the passage of time. In this regard, static analysis is superior.

While online services that evaluate malware samples by running antivirus products against them and reporting the results are useful in judging the accuracy of antivirus products, this practice does not constitute "malware analysis" and should not be used for evaluating a Trojan defense claim. In our direct, personal experience, antivirus software is frequently woefully inaccurate in either detecting or categorizing new malware samples. Sometimes antivirus simply fails to detect malware at all, and sometimes the malware is identified as a variant of existing malware with which it actually has no relationship at all.

Of course, analysis procedures to get to the bottom of whatever actions a malware sample might perform are useless if the malware isn't detected in the first place. Memory forensics offers both better detection and understanding of sophisticated malware.

5. Memory Forensics vs The Trojan Horse Defense

Memory forensics encompasses the set of techniques to first acquire and then analyze a sample, or snapshot, of physical memory at a particular moment in time. The dawn of memory forensics began when malware authors realized that through anti-forensics techniques they were able to bypass many or all of the traditional forensics analysis procedures. For example, security researchers documented that, contrary to popular belief, an application does not ever have to be written to disk to be executed. Publicly available research papers document memory-only execution of applications and libraries as far back as 2004 (Skape and Turkulainen, 2004; grugq, 2004). These techniques are now commonplace and implemented in popular open-source offensive security projects, such as Meterpreter (Wadner, 2014) and PowerShell Empire (Schroeder, 2019), as well as a wide variety of malware used during real attack campaigns.

Besides hiding executables in memory, malware can also be programmed to hide the rest of its presence on the live system. This can be accomplished via a technique known as API hooking, which allows malware to filter the types of information viewed and accessible by end users of the system, as well as other software running on the system. Common hiding techniques on Windows systems include hiding the malware's processes from Task Manager, files from Explorer, and network connections from *netstat*. Other tools to investigate system resources will similarly be affected by the hooking, and the malware will be hidden from them also. We note that the ability to run executables that reside only in memory and to hook APIs is widely abused on all major platforms, including Linux and Mac, and not just on Windows.

Beyond malware, other types of applications have similarly worked to avoid traditional digital forensics. The most prominent examples are applications that implement a "private" usage model, such as private browsing modes that do not log browsing history or cookies to disk, and chat applications that use end-to-end encryption on the network and do not keep local records of chat history. These efforts to specifically circumvent traditional forensics analysis have contributed to an increased awareness of the rising need for memory forensics and demands for ongoing advances in this field.

5.1. Techniques for Acquiring Memory

There are two categories of techniques for acquiring memory: software-based acquisition and hardware-based acquisition. Hardware-based acquisition was initially popular due to the fact that widely-accessible protocols, such as Firewire and PCMCIA, allowed direct access to physical memory by attaching hardware devices and using DMA. Specialized hardware devices take advantage of these protocols to acquire memory from systems after being attached. Hardware-based acquisition bypasses the need to log into systems, which requires valid credentials, or to load custom software. While initially popular, hardware-based acquisition is now less commonly used due to the inability of pervasive technologies such as USB devices to access all of physical memory, as well as operating system protections against DMA-based attacks. For example, Mac OS X systems prevent attached devices from performing DMA operations while the system is locked. Only after valid credentials are entered do the devices gain access. The requirement of valid credentials substantially reduces the power and scope of hardware-based mechanisms. Now, the use of software-based mechanisms is much more common.

Software-based acquisition utilizes kernel-level code to access all of physical memory. The memory contents are typically written to the file system of an external storage device or delivered securely over a network connection to a remote machine. Software-based acquisition has an advantage over hardware-based solutions in that it can successfully acquire memory from a variety of operating systems and hardware configurations, as it does not rely on specific hardware protocols being present or active. While widely supported, software-based acquisition does have two main drawbacks. First, to load a kernel driver, valid administrator credentials are required. This is generally not an issue in enterprise networks where the IT team manages user accounts, but is a major stumbling point in law enforcement operations when the suspect may not be legally required to reveal their credentials, or the operation is supposed to be covert and the suspect not alerted. Second, malware running on the system at the time of acquisition has the potential to interfere with the acquisition process. This capability has been demonstrated in many research projects and several malware samples found in the wild have implemented interference techniques. Fortunately, most attempts to interfere with the acquisition process are rather obvious, such as completely blocking acquisition when it would otherwise work, or tampering with in-memory data in a way that produces easy-to-detect discrepancies during analysis.

5.2. Analyzing Memory Samples

After a valid memory capture is made, the sample is then analyzed in a forensic lab using appropriate software. Relative to the Trojan horse defense, analysis of a memory sample provides numerous benefits, including the ability to detect stealthy malware, uncover precise user actions not accessible by traditional analysis techniques, and help prove intent, or lack thereof, on the part of the person under investigation.

5.2.1. Memory Forensics and Modern Malware

The true power of memory forensics is illuminated when applied to the investigation of modern malware. Since memory forensics provides the ability to examine the entirety of the state of a system, there is little room for malware to hide or act in a covert manner. This presents a significant advantage to the investigator as they can operate with increased confidence that any malware that is present will be found. In the case of memory-only malware, which is essentially invisible to traditional forensics techniques, memory forensics can provide three distinct benefits. First, it can detect that something is "off," based on the fact that active code and data structures in memory are not associated with a file on disk. This is not a situation that occurs using legitimate application development practices. Second, besides simply detecting that something malicious in on the system, tools can pinpoint precisely where the malicious code is active in memory and then automatically extract it for further scrutiny. This allows retrieval of code and data that was never actually written to the file system of the machine under investigation. Third, data that malware hides from live and traditional forensics methods, such as running processes and open network connections, are fully visible to memory forensics tools. Leveraging these benefits, memory forensics provides not only the ability to detect malware, but also mechanisms for deep understanding of malware behavior.

Volatility (Volatility Foundation, 2017) is the most widely used and powerful framework for memory forensics, containing an entire suite of analysis plugins dedicated to the GUI subsystem of Windows (Ligh, 2012a,b). This subsystem powers all interactions the end user has with the keyboard, mouse, monitor, and other devices. For malware to fake user keystrokes, such as typing in the URL of an illegal website, this subsystem must be used. To detect such behavior, Volatility has several analysis plugins that specifically look for code performing such actions. In situations where these plugins report active code, that code must then be investigated to determine its precise actions and reasons for controlling hardware devices. In some cases, actions such as the implementation of hot keys or custom mouse controls turn out to be legitimate, but in other cases, this analysis ends up pointing directly to the malware responsible for framing an innocent party.

Another type of analysis concerns malware programmed to contact a list of illegal websites from a victim's computer. When assessing whether the websites were accessed intentionally by the user or programmatically by malware, the list of accessed URLs can often be extracted from the acquired sample. This can potentially reveal many details to the investigator, including which applications were referencing the URLs; which files, if any, contained the URLs; and the context of the URL within applications, such as them appearing inside an advertisement shown in a browser, or in the copy/paste buffer on the user's system. By analyzing the full context of illegally accessed material, the investigator can build a strong and defensible case about the guilt or innocence of the person under investigation.

5.2.2. Anti-Forensics Applications

Investigating computer systems where anti-forensics techniques are in place can be extremely difficult. The purpose of anti-forensics tools is to either erase or scramble the digital artifacts investigators rely on to perform their work. In cases where the Trojan defense is employed, the use of anti-forensics can and does remove the exact artifacts relied upon during traditional analysis to tie activity back to either the end user or malware. Fortunately, memory forensics is not as easily affected by anti-forensics techniques. Even artifacts that have been mangled on disk can still be recoverable in memory. A common example of this is when a user securely deletes a file from disk using a third party wiping utility. For the subset of utilities that implements secure deletion properly, the file is truly gone and unrecoverable from non-volatile storage after the wiping operation completes. Using memory forensics techniques can circumvent such deletion, however, as remnants, or even a complete copy, of the file may remain in memory.

5.2.3. "Private" Applications

As mentioned previously, web browsers, chat programs, and other common applications routinely implement private usage modes. The purpose of these operational modes is to ensure that passive monitoring of an application's network traffic, as well as traditional forensic investigation of the system's hard drive, reveals little to no detail of the activity that occurred. This is problematic for all sides in a legal investigation. The lack of on-disk browser history can make it difficult for the investigator to tie visiting specific websites to the user, and inversely, difficult for the user to show that they did not visit those websites.

As with malware and anti-forensics applications, memory forensics can still recover most or all of the activity that occurred in these private usage modes. This is possible because all of the activity that happens in the applications - typing text, viewing pictures, sending and receiving chat messages, browsing web pages, choosing file names, and more - all leave traces in memory. In the case of end-to-end encrypted chats, which are only visible to the actual participants in the chat, entire plaintext copies of conversations will be left in memory. This occurs because memory buffers that are created after decryption to display the plaintext data in the chat window remain in memory.

Browsers suffer a similar fate under the scrutiny of memory forensics techniques, even for data that was sent and received using HTTPS. Again, for the browser to display a page's content in a readable form, the decrypted network traffic must be stored in memory. Similarly, before an encrypted request is made by the browser, the plaintext data must first be stored in memory for encryption. The data generated by applications, such as browsers and chat programs, often stays in memory long after the application is closed.

Many forensic tools exploit the gap between what traditional forensic analysis can uncover versus what memory forensics can uncover. Bulk Extractor (Garfinkel, 2013) is one of the most widely used, analyzing every byte of a memory capture to look for data, including URLs, emails, DNS lookups, social media usage, web searches, and other artifacts. By simply running Bulk Extractor against a memory sample, investigators can uncover a wealth of forensic data that may be inaccessible to traditional digital forensics.

5.2.4. Encrypted Stores

The use of encryption is now widespread, even for novice users. Mainstream operating systems provide full disk or partition encryption by default and this is now the expectation when configuring a new computer. Windows, Linux, and Mac all also natively support creation of encrypted containers. These containers are stored as a single file on disk, but then internally have an encrypted file system. The use of containers is very attractive as they can easily be moved to different devices using email, removable storage devices, or cloud storage. They are also useful in that a user's sensitive files can be enveloped by an extra layer of protection beyond full disk encryption. Of course, criminals are well aware of these encrypted containers and use them to store information they would prefer law enforcement and digital investigators not to find.

When a system is examined, investigators will often use forensic software that is designed to find encrypted containers. Finding the containers is only the first step, however; the credentials to unlock them must be obtained. In countries where users do not have to divulge their credentials to law enforcement, this can immediately derail an investigation. Fortunately, memory forensics can be used to gain access to encrypted containers in cases where the credentials are unknown. By focusing on keystrokes, such as the user typing in their password(s), as well as the contents of open files, including text files of passwords or a password manager's database, tools can still access protected information in memory. This might allow an investigator to build a password-cracking dictionary that is closely tailored to the user under investigation. In many cases where encrypted containers are present, the use of a memory sample to crack the container is the difference between a guilty person going to jail or walking free.

6. Conclusion

This chapter has considered the Trojan horse defense and whether traditional forensics techniques are sufficient to definitively detect cutting-edge malware and disentangle the actions of legitimate users from those covertly executed by malware. In general, the answer is no. Modern malware is often weaponized, extremely stealthy, and in many documented cases, leaves no trace on non-volatile storage. Furthermore, modern malware can secretly perform virtually any action a user might perform on a computer system, including actions that place human users in serious legal jeopardy, such as launching attacks against other systems or downloading illicit materials. Traditional approaches to detecting malware that might impact a legal case include executing antivirus software and using "pull the plug" forensic techniques that target only non-volatile storage devices. Unfortunately, in many cases when investigators restrict themselves to these techniques, they will be completely unable to detect malware, much less establish whether malware played a substantial role. When malware is in fact responsible for illegal actions, this choice of tools tips the scales heavily against innocent parties.

Memory forensics is already widely adopted in digital forensic and incident response teams in both the private and the public sphere, as it often provides the sole viable solution for detecting and understanding modern malware, especially strains that are memory-only or file-less. Organizations have rushed to train their investigative teams in how to use memory forensics and put processes in place to increase the likelihood of success in real investigations. While the use of memory forensics tools currently require substantial expertise, improving their reliability and usability is an area of active research. Unfortunately, the use of memory forensics is not particularly widespread in the law enforcement community, especially in smaller, local organizations and departments. Given the power of memory forensics to provide investigators with viable tools for properly detecting and analyzing modern malware, memory forensics simply must become part of the modern digital forensics toolkit used by law enforcement.

7. Acknowledgments

The authors would like to thank LSU Law Professor Ken Levy for helpful comments on early versions of this chapter.

References

AccessData (2019). AccessData FTK Imager. `https://marketing.accessdata.com/ftkimager4.2.0`.

AccessData (2019). Forensic Toolkit (FTK). `https://accessdata.com/products-services/forensic-toolkit-ftk`.

Alperovitch, D. (2016). Bears in the Midst: Intrusion into the Democratic National Committee. `https://www.crowdstrike.com/blog/bears-midst-intrusion-democratic-national-committee/`.

Armerding, T. (2013). Advanced Volatile Threat: New Name for Old Malware Technique. `http://www.csoonline.com/article/2132995/malware-cybercrime/advanced-volatile-threat--new-name-for-old-malware-technique-.html`.

AV-Test (2019). Malware. `https://www.av-test.org/en/statistics/malware/`.

Blackbag Technologies (2019). Blacklight. `https://www.blackbagtech.com/blacklight.html`.

Bowles, S. and Hernandez-Castro, J. (2015). The First 10 Years of the Trojan Horse Defence. *Computer Fraud & Security*.

Brenner, S. W., Carrier, B., and Henninger, J. (2004). The Trojan Horse Defense in Cybercrime Cases. *Santa Clara High Technology Law Journal*, 21.

Carrier, B. (2003). Defining Digital Forensic Examination and Analysis Tools using Abstraction Layers. *International Journal of Digital Evidence*, 1(4):1–12.

Carvey, H. (2019). RegRipper. `https://github.com/keydet89/RegRipper2.8`.

Cuckoo Foundation (2014-2019). Cuckoo Sandbox Automated Malware Analysis. `https://cuckoosandbox.org`.

Cybercrime (2019). Bureau of Justice Statistics. `https://www.bjs.gov/index.cfm?ty=tp&tid=41`.

DEEPSPAR Data Recovery Systems (2019). PC-3000 Data Extractor. `http://www.deepspar.com/products-pc-3000-data.html`.

Doffman, Z. (2019). Warning As Devious New Android Malware Hides In Fake Adobe Flash Player Installations (Updated). `https://www.forbes.com/sites/zakdoffman/2019/08/16/dangerous-new-android-trojan-hides-from-malware-researchers-and-taunts-them-on-twitter/#70c2245c6d9c`.

Eckelberry, A., Dardick, G., Folkerts, J. A., Shipp, A., Sites, E., Stewart, J., and Stuart, R. (2007). Technical Review of the Trial Testimony State of Connecticut vs. Julie Amero. `https://web.archive.org/web/20090124121203/http://www.sunbelt-software.com/ihs/alex/julieamerosummary.pdf`.

Fortuna, A. (2017). Malware Persistence Techniques. `https://www.andreafortuna.org/2017/07/06/malware-persistence-techniques/`.

Garfinkel, S. L. (2013). Digital Media Triage with Bulk Data Analysis and bulk_extractor. *Computers & Security*, 32:56–72.

Ghavalas, B. and Philips, A. (2005). Trojan Defence: A Forensic View Part II. *Digital Investigation*, 2.

Grier, J. and Richard, G. G. I. (2015). Rapid Forensic Imaging of Large Disks with Sifting Collectors. *Digital Investigation*, 14:S34–S44.

grugq (2004). FIST! FIST! FIST! It's All in the Wrist: Remote Exec. `http://phrack.org/issues/62/8.html#article`.

Haagman, D. and Ghavalas, B. (2005). Trojan Defence: A Forensic View. *Digital Investigation*, 2.

Kaspersky Research Team (2014). Kaspersky Lab Uncovers "The Mask": One of the Most Advanced Global Cyber-espionage Operations to Date Due to the Complexity of the Toolset Used by the Attackers. `https://usa.kaspersky.com/about/press-releases/2014`.

Kaspersky Research Team (2015). The Mystery of Duqu 2.0: A Sophisticated Cyberespionage Actor Returns. `https://securelist.com/the-mystery-of-duqu-2-0-a-sophisticated-cyberespionage-actor-returns/70504/`.

Laing, B. (2018). Malware-as-a-Service: The 9-to-5 of Organized Cybercrime. `\sloppyhttps://www.lastline.com/blog/malware-as-a-service-the-9-to-5-of-organized-cybercrime/`.

Ligh, M. H. (2012a). MoVP 3.1 Detecting Malware Hooks in the Windows GUI Subsystem. `https://volatility-labs.blogspot.com/2012/09/movp-31-detecting-malware-hooks-in.html`.

Ligh, M. H. (2012b). OMFW 2012: Malware In the Windows GUI Subsystem. `https://volatility-labs.blogspot.com/2012/10/omfw-2012-malware-in-windows-gui.html`.

Luehr, P. H. (2005). Real Evidence Virtual Crimes. *Criminal Justice*.

Majumder, B. G. (2019). New Malware Infects Thousands of PCs, Confirms Microsoft and Cisco Talos. `https://www.ibtimes.sg/new-malware-infects-thousands-pcs-confirms-microsoft-cisco-talos-32615`.

Malin, C. H., Casey, E., and Aquilina, J. M. (2008). *Malware Forensics: Investigating and Analyzing Malicious Code*. Syngress.

Malwarebytes Labs (2019). 2019 State of Malware. `https://resources.malwarebytes.com/resource/2019-state-malware-malwarebytes-labs-report/?utm_source=blog&utm_medium=post&utm_campaign=0119_ws_stateofmalwarereportq119_mb`.

Mcelroy, W. (2007). In Child Porn Case, Technology Entraps the Innocent. `http://www.independent.org/news/article.asp?id=1894`.

Microsoft (2011). What is Antivirus Software? `http://web.archive.org/web/20110411203211/http://www.microsoft.com/security/resources/antivirus-whatis.aspx`.

Monterosso, F. S. (2010). Protecting the Children: Challenges that Result in, and Consequences Resulting From, Inconsistent Prosecution of Child Pornography Cases in a Technical World. *XVI Rich. J.L. & Tech.*, 11.

O'Donnell, L. (2019). Joker Spyware Found in 24 Google Play Apps. `https://threatpost.com/joker-spyware-google-play-apps/148053/`.

OpenText Security (2019). Encase Frorensic. `https://www.guidancesoftware.com/encase-forensic`.

PCWorld (2008). A Misconfigured Laptop, A Wrecked Life. `https://abcnews.go.com/Technology/PCWorld/story?id=5188541`.

Rankin, B. (2018). A Brief History of Malware Its Evolution and Impact. `https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/`.

Richard, G. G. and Roussev, V. (2005). Scalpel: A Frugal, High Performance File Carver. *Digital Forensics Research Conference (DFRWS)*, pages 71–77.

Schroeder, W. (2019). PowerShell Empire. `https://www.powershellempire.com/`.

Schwartz, E. J., Avgerinos, T., and Brumley, D. (2010). All You Ever Wanted to Know About Dynamic Taint Analysis and Forward Symbolic Execution (But Might Have Been Afraid to Ask). In *2010 IEEE Symposium on Security and Privacy*, pages 317–331. IEEE.

Skape and Turkulainen, J. (2004). Remote Library Injection. `http://www.hick.org/code/skape/papers/remote-library-injection.pdf`.

Steel, C. (2014). Technical Soddi Defenses: The Trojan Horse Defense Revisited. *Journal of Digital Forensics, Security and Law.*

Symantec (2018). Internet Security Threat Report Volume 23. `https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf`.

Volatility Foundation (2017). The Volatility Framework: Volatile Memory Artifact Extraction Utility Framework. https://github.com/volatilityfoundation/volatility.

Wadner, K. (2014). An Analysis of Meterpreter during Post-Exploitation. https://www.sans.org/reading-room/whitepapers/forensics/paper/35537.