EVALUATION OF MALWARE TARGET RECOGNITION DEPLOYED IN A CLOUD-BASED FILESERVER ENVIRONMENT

THESIS

G. Parks Masters, Second Lieutenant, USAF

AFIT/GCO/ENG/12-08

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

AFIT/GCO/ENG/12-08

EVALUATION OF MALWARE TARGET RECOGNITION DEPLOYED IN A CLOUD-BASED FILESERVER ENVIRONMENT

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Air Force Institute of Technology - Graduate School of Engineering and Management

Air Force Insitute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science

G. Parks Masters, B.S.C.S.

Second Lieutenant, USAF

March 2012

AFIT/GCO/ENG/12-08

EVALUATION OF MALWARE TARGET RECOGNITION DEPLOYED IN A CLOUD-BASED FILESERVER ENVIRONMENT

G. Parks Masters, B.S.C.S.
Second Lieutenant, USAF

Approved:

Barry E. Mullins, PhD (Chairman)	Date
Maj Thomas E. Dube, PhD (Committee Member)	Date
Richard A. Raines, PhD (Committee Member)	Date

Abstract

Cloud computing, or the migration of computing resources from the end user to remotely managed locations where they can be purchased on-demand, presents several new and unique security challenges. One of these challenges is how to efficiently detect malware amongst files possibly stored in multiple locations across the Internet over congested network connections. This research studies how such an environment will impact performance of malware detection.

A simplified cloud environment is created where network conditions are fully controlled. This environment includes a fileserver, a detection server, the detection mechanism, and clean and malicious file sample sets. The performance of a novel malware detection algorithm called Malware Target Recognition (MaTR) is evaluated and compared with several commercial antivirus applications at various congestion levels. MaTR is an engineering proof-of-concept prototype and has not been multithreaded or optimized for performance as the commercial products have been. This research evaluates performance in terms of file response time and detection accuracy rates.

Although MaTR demonstrates competitive response times at lower congestion levels, when severe packet loss is introduced MaTR's response times are slowed by a factor of up to 817 for clean files and 334 for malicious files. Commercial products are slowed at most by a factor of 16 for malicious files and 137 for clean files often outperforming MaTR. MaTR's true mean response time when scanning clean files with low to moderate levels of congestion is roughly 0.05s, which is not statistically significantly different than leading commercial response times. MaTR demonstrates a slightly faster response time, by roughly 0.1s to 0.2s, at detecting malware at these congestion levels, but MaTR is also the only device that exhibits false positives with a 0.3% false positive rate. This is expected as MaTR is the only non-signature based detection mechanism. MaTR's true positive detection rates are extremely competitive at 99.1%.

Acknowledgments

I would like to thank my thesis advisor, Dr. Barry Mullins, for his exceptional guidance and encouragement during my research endeavors. His expert balance of keeping me on course and allowing me to pursue various research ideas at my own pace created a productive work environment and minimized stress while accomplishing this research. I would also like to thank Maj Thomas Dube for his assistance in implementing and troubleshooting his Malware Target Recognition (MaTR) software.

G. Parks Masters

Table of Contents

List of Figures

List of Tables

Evaluation of Malware Target Recognition Deployed in a Cloud-Based Fileserver Environment

1 Introduction

Technology and electronics are gradually becoming dependent on a cloud computing model. While there are many definitions of cloud computing, it basically describes the migration of computing resources from the end user to remotely-managed locations where these resources can be accessed or purchased on demand [Mic09]. Commonly used applications such as Gmail, Dropbox, and even Facebook all implement forms of cloud computing as they rely on remote resources that consumers do not have direct access or control. This transition to relying on computing resources that exist in remote, often unknown locations presents several unique security challenges. This research addresses part of the challenge of how to detect malware on fileservers offered as a cloud-based service.

In a cloud fileserver environment, users perceive their files to be stored in one central location when, in reality, they could be located on multiple hosts across the Internet. The distributed storage of files, network congestion, and sheer volume of files, can severely impact the performance of traditional malware detection techniques. Commercial antivirus scanners place significant computational resource demands on modern computers when scanning local files. The purpose of this research is to determine how a cloud fileserver environment will impact the performance of remote malware detection. Specifically, this research observes the execution of different malware detection mechanisms when scanning files in a remote location with varying levels of congestion. This research conducts performance evaluation of commercial antivirus products and a

generic detection algorithm called Malware Target Recognition (MaTR). MaTR demonstrates promising results in initial testing and has the potential to be highly effective in a cloud-based fileserver [DRP+12][Mer11].

The goals of this research are to emulate a simplified cloud fileserver environment and evaluate file response times and detection accuracies. The version of MaTR used in the experiments requires the entire file to be loaded into memory, which involves transferring it across a potentially congested network. As such, this research hypothesizes that MaTR's file response times will be marginalized and will not outperform the commercial detection mechanisms. This research also hypothesizes that congestion rates will not impact detection rates, and MaTR continues outperform other detection mechanisms in detection accuracy rates as it has demonstrated in past experiments [DRP+12].

Chapter 2 of this thesis presents existing literature and research related to the problem described above. Cloud computing is defined and file storage of a cloud based service is discussed. Modern malware detection techniques are described along with the MaTR algorithm. Some related work regarding malware detection as a cloud-based service is also discussed. Chapter 2 concludes by synthesizing some of the information in the literature review to provide support for the underlying assumptions and decisions made during the experimental design.

Chapter 3 discusses the methodology for the experimental design. This includes a more in-depth problem definition, detailed goals of this research, hypotheses for expected research outcomes, and the approach to accomplishing the goals. This chapter also provides details on how the cloud environment is emulated and the experiments are conducted. Lastly, Chapter 3 presents how the results of the experiments will be statistically analyzed.

Chapter 4 presents the results of the experiments. These include analysis of system response time and detection rates. These results are compared between different mechanisms at different factor levels, and an interpretation of the results is provided.

Chapter 5 provides the conclusion to this research. This includes a synopsis of the research conducted, a summary of the results, and key conclusions that are drawn from those results. This chapter also contains recommendations for future work and potential follow on research.

2 Literature Review

This chapter describes relevant research relating to malware detection in a cloud fileserver environment. Section 2.1 defines cloud computing and outlines essential characteristics of a cloud computing environment. Section 2.2 discusses cloud-based fileservers to include key features of commercial implementations. Section 2.3 covers current methods of detecting malware, including signature-based and anomaly-based detection. Section 2.4 discusses some related research involving antivirus as a service hosted in the cloud. Lastly, Section 2.5 extends the literature review by synthesizing some of the research to provide a framework for the experimental design in Chapter 3.

Existing malware detection research for a cloud-based environment is not present in this review as it does not currently exist. Modern malware detection technologies primarily focus on protecting individual clients. This makes sense as files usually do not execute on a fileserver and are often stored locally. However, with the evolution of cloud computing and the migration of computer resources into the cloud, one must consider the need and potential benefits of scanning executables hosted on a cloud fileserver.

2.1 Cloud Computing Defined

The term "cloud computing" is used rather loosely in industry and social settings, and for good reason. With the Internet's ubiquity in modern living, many argue that some level of cloud computing is now a common occurrence. This research heavily focuses on cloud computing technology, and thus requires a formal definition of cloud computing. The National Institute of Standards and Technology (NIST) defines cloud computing as follows:

> Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly

provisioned and released with minimal management effort or service provider interaction [MG11].

The NIST definition also includes five essential characteristics, three service models, and four deployment models which are discussed later in this chapter. While this definition is somewhat complex, it simply describes many aspects of the Internet and the migration of computer resources from the end user to remotely-managed locations where they can be purchased on-demand [Mic09]. There are several well-known examples of cloud computing in use today including webmail providers, remote fileservers such as Dropbox, and Google Docs. Figure 2.1 provides a visualization of cloud computing and some of its common applications.

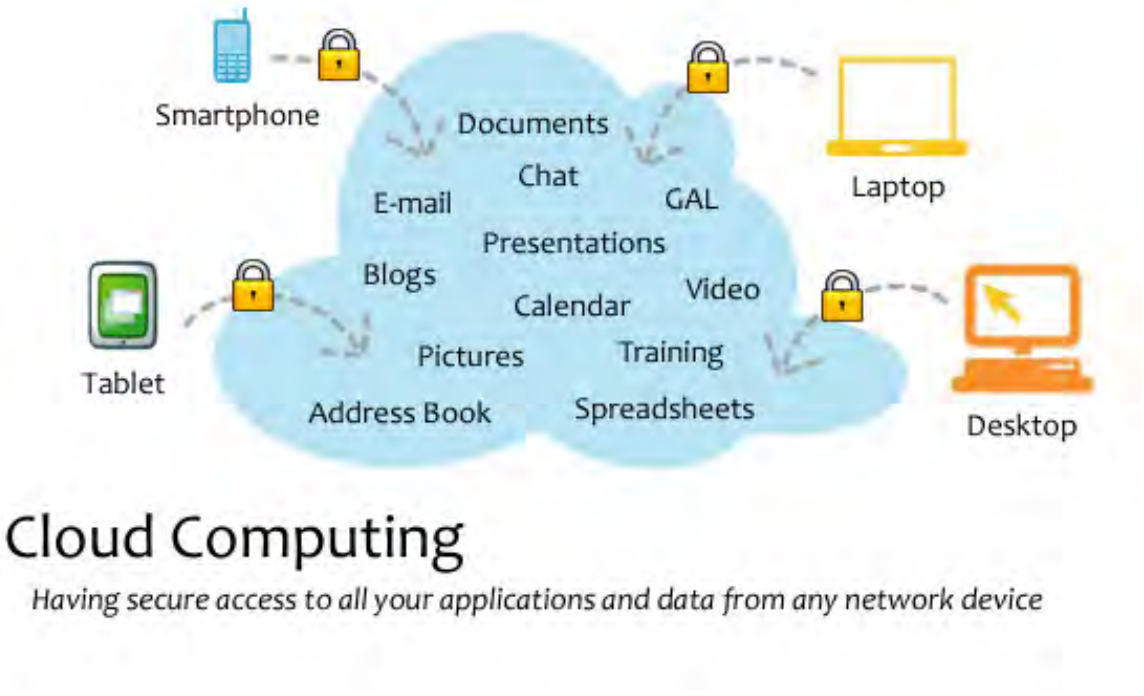

Figure 2.1: Cloud Computing Visual Diagram [GB11]

Amazon offers a clear example of cloud computing [Ser11]. Their service, called Amazon Elastic Compute Cloud (EC2), allows consumers to buy computing resources to run servers or whatever software they like. EC2 is rapidly scalable, both up and down, highly configurable, and they only charge consumers for what they use. In many cases this approach can be significantly cheaper than a company buying and maintaining local server hardware and resources [Ser11].

This research must accurately model a cloud computing environment, and as such the key characteristics that make such an environment must be defined. The NIST definition also provides the essential characteristics, service models, and deployment models of cloud computing.

2.1.1 Essential Characteristics. This section describes the essential characteristics that are included as part of NIST's definition of cloud computing [MG11].

- *On-demand self-service*. Consumers can easily acquire or purchase additional computing resources such as server time and network storage. This capability should be available on demand without human interaction from the provider.

- *Broad network access*. Access to cloud capabilities is available over the network. This access should be compatible with a wide variety of network devices such as smart phones, laptops, and PDAs.

- *Resource pooling*. The provider's computing resources are pooled so that can be physically or virtually assigned and reassigned to consumers on demand. The consumer generally is unaware of, and has little control over, where the computing resources physically exist. Some of these pooled resources may included storage, processing, memory, network bandwidth, and virtual machines.

- *Rapid elasticity*. Capabilities are rapidly and elastically provided to the consumer as needed. From the consumer's perspective, resources appear unlimited and are available for purchase at any time.

- *Measured Service*. Cloud services are optimized by providing a gauge of resource consumption at some level of abstraction. Resources measured include storage, processing, and bandwidth. This ability allows resources to be monitored, controlled and reported. This information is used by both the provider and the consumer to provide transparency into the utilized service.

These characteristics are later evaluated to ensure this research accurately emulates a cloud environment.

2.1.2 Service Models. There are three different service models as described by the NIST, which are briefly described below. These descriptions are used to identify which model this research falls under [MG11].

- *Cloud Software as a Service (SaaS).* This service model includes capabilities provided to the consumer to use applications running on the provider's cloud infrastructure. Access to these applications is provided through a thin client, such as a web browser or another application possessed by the consumer. The consumer does not manage or control the underlying cloud infrastructure beyond possible configuration settings within the cloud software. An example of this service model is Gmail.

- *Cloud Platform as a Service (PaaS).* This service model provides consumers with the capability to deploy their own software onto the cloud infrastructure. The consumer does not manage the underlying infrastructure such as the operating systems, network, or other computing resources. The software deployed must be supported by the cloud environment hosting the application, often requiring

applications to be developed in certain programming languages or with specific tools. The consumer controls the deployed applications and possibly configuration settings of the hosting environment. For example, a consumer desires to perform computationally-intensive task such as protein folding. A cloud platform service may be purchased, and the consumer can deploy his protein folding application to the platform which consumes cloud computing resources.

- *Cloud Infrastructure as a Service (IaaS).* This service provides the consumer with the capability to provision fundamental computing resources such as processing, storage and networks. This offers the consumer the ability to deploy and run arbitrary software including operating systems and applications. The consumer still does not control the underlying cloud infrastructure. An example of this is a consumer purchasing a virtualized server operating in the cloud.

A cloud-based fileserver falls under either SaaS or IaaS depending on how the fileserver is configured. An implementation such as Dropbox employs a software front-end accessible through a web browser with a cloud infrastructure as the back-end. Consumers do not have direct access or control to the infrastructure hosting Dropbox, whose implementation follows the SaaS model. A fileserver could also be deployed on a fully functioning virtualized server operating in the cloud, which would be Infrastructure as a Service model. This research conducts experiments that emulate the IaaS model. A fully functioning fileserver is created on virtualized hardware and accessed across a network that emulates the Internet.

2.2 File Storage as a Cloud-based Service

This section discusses several popular commercial implementations of a cloud-based fileserver. The operation of each implementation is explained as well as the security measures they offer.

2.2.1 Dropbox. Dropbox, founded in 2007, offers users 2GB of free online storage which can be accessed from anywhere with an Internet connection. It offers a simple way to sync files amongst multiple computers, contains features to easily share files and folders in your Dropbox with others, and may be used as a file backup tool. Additional storage of up to 100GB may be purchased [Dro11].

Dropbox creates a folder on the computer upon installation. Anything placed in that folder is synchronized to Dropbox's cloud fileserver as well as all other computers that have the client installed. This provides a simple drag-and-drop interface to upload files to the cloud [Dro11].

2.2.1.1 Unique Features. The following items are unique features that dropbox provides to consumers [Dro11].

- Files are synced to the computer's hard drive making them available offline.
- Dropbox works with all operating systems as well as mobile devices.
- Dropbox only transfers the parts of files that change as opposed to re-uploading the entire file upon each change.
- Before a file is uploaded, the Dropbox client or web interface first creates a hash of the file. It compares the hash with the hash of every other file in existence on the Dropbox servers. If an exact duplicate of the file already exists somewhere in the Dropbox cloud, it is simply copied locally from one location on the fileserver to the location hosting the consumers files. This is generally much faster, eliminating the need for some files to be uploaded over a potentially congested network.

2.2.1.2 Security Features. The following items are security features that are provided by Dropbox [Dro11].

- Dropbox keeps a one-month history of items uploaded so they can be easily recovered if accidently deleted.
- Dropbox transmits files to and from the cloud over an encrypted channel.
- All files stored with Dropbox are encrypted in the cloud and cannot be viewed by employees or others unless explicitly shared.

2.2.2 Egnyte Cloud Fileserver. Egnyte is another cloud-based fileserver that supports corporate environments. Egnyte eliminates the need to purchase traditional fileservers, backup mechanisms, and File Transfer Protocol (FTP) solutions, saving the company money in the long run. As opposed to Dropbox, they offer support for much larger storage capacities and large file sizes [Egn11].

2.2.2.1 Unique Features. The following items are unique features provided by Egnyte [Egn11].

- Egnyte employs a hybrid-cloud mechanism that involves network-attached storage containing a copy of the cloud contents on the local network to allow access to the files with Local Area Network (LAN) speeds. These documents are then synced to the cloud for additional backup. This also provides access when the local network loses outbound connectivity.
- Egnyte also offers a personal local cloud when the hybrid-cloud approach is not available. This simply stores the files on the local computer or directly attached storage, and synchronizes the documents to the cloud from there.
- Sharing is simplified by adding users to an "Address Book" for a particular file or folder. When uploaded to the cloud, the users in the address book will have access to the appropriate files. This feature also includes access control permissions such as read or write.

2.2.2.2 Security Features. Egnyte encrypts all transfers to and from the cloud. However, it does not encrypt the data on the cloud fileserver. Instead, the implementation has strict access controls verifying usernames and passwords on each access. Egnyte also provides consumers with descriptions of the robust security mechanisms in place at their storage facility and guarantee the security of the data held within [Egn11].

2.2.3 Commercial Cloud-based Fileserver Summary. Commercial cloud-based fileserver solutions are becoming popular on the Internet (along with other commercialized cloud-computing resources). Many other implementations exist with similar features. For example, box.com offers the ability to edit or view known file types through a web interface. This includes editing Word and Excel documents or watching videos straight from the browser without having to download the files.

An important aspect of these fileserver solutions is the restriction to read/write permissions on the cloud infrastructure. A file cannot be executed *on* a cloud fileserver. Instead, it must be downloaded and executed locally. This is an important characteristic that this research emulates during experimental design.

2.3 Detecting Malware

This thesis uses the term malware to describe a broad range of "malicious software" categories. These can include viruses, Trojans, dialers, backdoors, and exploits. Christodorescu et al. define malware simply as a program designed with malicious intent [CJS+05]. Over time, malware complexity has continued to increase, with advanced obfuscation and packing techniques to hide from antivirus (AV) scanners, anti-debugging and anti-disassembling mechanisms to thwart analysis attempts, and a wide array of attack vectors to bypass intrusion detection systems. Likewise, antivirus software continues to become equally complex to achieve continued protection.

This research focuses on detecting Windows Portable Executable (PE) files. As such, the focus of the research reviewed also deals with detecting PE files. The two primary approaches for detecting malicious software are dynamic and static analysis (or a "hybrid" approach combining both) [IM07]. Dynamic analysis involves examining code with CPU emulation or while it is executing. In the context of this research, this is impossible due to the permission settings of the emulated cloud-based fileserver (read/write only). Therefore, this literature review focuses on static analysis techniques.

A malware detector is simply the application of a malware detection technique. According to [IM07], detectors take two inputs: the executable in question and some type of knowledge of either malicious behavior or normal behavior.

2.3.1 Static Analysis. Static analysis interrogates an executable's syntax or analyzes internal structure properties to determine whether it is malicious. Within the realm of static analysis, detection is accomplished through either observing anomalies usually absent in benign files or via signatures matching pre-existing malware [IM07].

2.3.1.1 Anomaly-based Detection. Anomaly detection attempts to determine known good characteristics of executables, and identifies malware based on divergences from that knowledge. In the context of the malware detector, the second input is knowledge of *known good behavior and valid file structures*. There are generally two phases to anomaly detection, a training or learning phase, and a detection phase. In static anomaly detection, the detector uses characteristics about the file structure of the program to detect malware. A key advantage to anomaly-based detection is that it does not require execution of the potentially malicious code [IM07].

Stolfo et al. [SWL07] presents a new detection mechanism for stealthy malware that is embedded in other files as an augmentation to existing Antivirus (AV) software. An example of this is malicious code embedded in a PDF document. This type of malware is

one example that effectively thwarts signature-based detection mechanisms, and thus poses a great threat as stealthy malware. Stolfo's technique involves a statistical binary content analysis of n-grams (simple byte sequences). When a file is scanned, the n-gram results are compared against known good n-grams of the same file type which the authors call fileprints. A mismatch indicating anomalous file sections could indicate the presence of malware embedded in the file requiring further examination. Their research focuses on malware that has been injected into either the head or the tail of the file. When malware is injected into the middle, this detection technique is less effective. Their experiments achieved between 72% and 94.5% detection rates of embedded malware depending on the malware's position in the file and size of the n-grams. As this method works regardless of the malware's contents, it excels at detecting novel malware where signature-based detection fails. A limitation of the technique is that it is based on a static snapshot of the training set, which could be an irrelevant sampling. Stolfo et al. also conduct experiments to determine whether this technique is capable of distinguishing malware from "normal" Windows executable fileprints, and to establish whether the technique is effective in identifying malware that has been packed or obfuscated. The results from these experiments vary widely and need further research to determine whether n-gram analysis is effective in those applications.

Shafiq et al. [SKF08] continue Stolfo's research on n-gram analysis. They implement conditional, or Markov n-gram, as opposed to the previous research's traditional n-grams. After a scan, the Markov n-gram's entropy rate is compared to that of the known good file type. Their research shows that malware injections significantly perturb the entropy rate of the known good file, which is how the malware is identified. The results of the experiment shows that this technique is much more effective at detecting malware than the traditional n-gram technique. Furthermore, their approach is able to detect malware injected anywhere in the file, as opposed just the head or tail of the file as with Stolfo et al.

[SWL07]. The Markov n-gram analysis also reduces the false positive rate of the previous technique by up to 48%. Still, the false positive rates are much higher than traditional AV software, with rates as high as 32%. As such, these techniques are best used to categorize malware as suspicious, and should be used alongside a commercial AV product.

In [Erd04], Erdelyi argues that the most reliable way of detecting stealth malware is through clean booting. Clean booting involves booting the computer with as few operating system components and drivers as possible, such as a DOS prompt. This simplifies detection of malware that hides itself. However, this is becoming more difficult with modern operating systems, such as NT-based systems, that still load a few basic drivers even when booting into safe-mode. If one of these drivers contains malicious code, it is capable of hiding itself at the kernel level. This is another form of anomaly-based detection as it uses a clean boot as knowledge of "good behavior".

Weber et al. [WSSG02] creates a tool called Portable Executable Analysis Toolkit (PEAT) designed to detect the presence of malware attached to executables. The toolkit implements three types of detection categories: 1) simple static checks, 2) visualization, and 3) statistical analysis. Static checks analyze the code for anomalies. For example, observing an entry point in an unusual section might suggest that malware is present. The visualization analysis uses graphical representations of the executable to detect anomalous and potentially harmful code. These include viewing ASCII strings, program disassembly results, and pictoral representations of memory access views. Finally, statistical tests such as instruction frequencies and patterns, register offsets, etc. are performed to discover anomalous code. This tool functions more as an analysis tool than an automated detector and requires an experienced operator to detect malware.

This research focuses on evaluating the performance of a PE malware detector called Malware Target Recognition, or MaTR [DRP+12]. Dube et al. use a decision tree learning algorithm and static heuristic features to discover malware. Their method does not rely on

a pristine disassembly, but instead uses high-level program information and common anomalies for malware detection. Dube et al. combine several of the anomaly-based static analysis techniques described above, as well as incorporate several of their own techniques. Early experimental results are very promising. Using large data sets of confirmed malware and non-malicious executables, MaTR demonstrates above a 99% accuracy rate with false positive and false negative rates less than 1/10%. MaTR also excels at detecting novel malware. It achieves a 99% detection rate which is vastly superior to the 60% detection rate a union of three commercial AV products achieve. Furthermore, MaTR is excellent at detecting obfuscated and packed malware; the presence of these defensive mechanisms alone often indicates malware. Early experiments show MaTR to be highly efficient with respect to runtimes. The average runtime to scan 278 samples is only 0.9s, whereas the runtime results on the same data set from three commercial AV products were 43s, 56s and 391s [DRP+12].

2.3.1.2 Signature-based Detection. Signature-based detectors require known malicious behavior as their input [IM07]. As the name suggests, characterizing malicious behavior, or creating signatures is the key to this method's success. This is the most common mechanism used by AV products today. However, new malware as well as slightly obfuscated old malware can often bypass these signature-based detection schemes [CJS+05].

Schultz et al. [SEZS01] were one of the first to present a technique for discovering *new* malware through a data mining framework. Their research method uses extracted data features including Dynamic Link Libraries (DLL) used, DLL function calls, and strings from existing malware and creates signature files. When scanning an unknown file, the framework looks for similarities with existing malware. Various experiments resulted in accuracy rates ranging from 83% to 97%, all significantly higher than the traditional signature technique which had an accuracy of just under 50%. The authors do note that

should the implementation details of the detection framework be compromised, many of these detection techniques could be evaded with simple obfuscation.

Kolter and Maloof extend Schultz's research in [KM04] and [KM06]. Their research distinguishes malware from benign files as well as classifies malware based on its payload. They use n-grams from large sets of benign and malicious executables, and use various inductive methods including naive Bayes, decision trees, support vector machines, and boosting. Boosted decision trees perform the best with a true positive rate of 98% and false positive rate of 5%.

Henchiri and Japkowicz further explore data mining capabilities for signature-based detection in [HJ06]. Their goal is applying machine learning to improve the problem of feature selection. They conduct an exhaustive search on a large set of known malware to determine some generic features. Then, the features that are most representative of malicious code are collected and used as the signature. Their research also evaluates the predictive power of a classifier by examining malware dependence relationships. A cross-validation scheme is used in an experiment to simulate a real-world virus outbreak. Depending on the feature set used, they achieve some promising results with up to 94% accuracy rates.

Sung et al. create an algorithm called SAVE (Static Analyzer of Vicious Executables) which attempts to detect known malware that is metamorphic, polymorphic, or has been obfuscated [SXCM04]. The authors argue that all versions of the same malware share a core signature that is a combination of several features from the code. The features they explore in their research are system calls. Sung et al. argue that even if a malware sample is obfuscated or restructured, the set of system calls is similar. As such, similarity measurements including cosine, Jaccard, and Pearson's correlation measures are taken between a sample under inspection and malicious signatures. The experiment is composed of only 20 samples, but SAVE successfully detects all of them. Further research with

larger sample sets and obfuscation techniques is required to determine SAVE's true effectiveness.

Christodorescu et al. conducts a series of research endeavors on signature-based static analysis and detection with comparisons to commercial AV products [CJ03] [CJ04] [CJS+05]. Their research uses control flow graphs (generated in IDA pro) as detection signatures in order to defeat common obfuscation techniques such as inserting *nops* (assembly code instruction for no operation) or code transposition which they show to thwart many antivirus techniques. They develop a proof-of-concept tool called SAFE (Static Analyzer For Executables) which is designed to resilient against these obfuscation techniques. In a limited experiment, SAFE detects all obfuscated malware.

Their method suffers from two key drawbacks. First, the overhead of generating and comparing control flow graphs can be substantial. The authors believe the observed performance is insufficient malware detection capacity in operational environments, especially when analyzing large executables. The second major drawback is packing. While the authors discuss obfuscation, as a static analyzer, SAFE is likely unable to generate a control-flow graph given a packed executable, a rather simple obfuscation technique [CJ04]. The authors extend this research in [CJS+05] using similar, refined techniques and achieve more reliable results in detecting malware. However, scan runtime performance is still a major point of concern.

2.4 Related Work

2.4.1 Antivirus as a Cloud-based Service. AV software has a reputation for negatively impacting users due to the software's intense resource demands. These demands stem from growing signature files caused by the constant influx of new malware samples, the rising time involved with de-obfuscating and unpacking binaries, and the continually increasing volume and size of files. Likewise, AV products require frequent signature updates, burdening bandwidth resources and making it less likely consumers are

up to date [YA09]. As these factors continue to hinder host-based AV efficiency, an alternative solution based in the cloud becomes more attractive. Several antivirus vendors, such as Panda Security, have already shifted their approach to a cloud-client architecture and call it the "next generation" of cloud computing [Res07].

The fundamental concept of AV in the cloud is that the complex analysis and detection mechanisms are relocated from the end user into the cloud. The traditional malware detector is replaced with a lightweight client that is responsible for capturing executables entering the host system and sending them to the cloud for further analysis [OCJ07]. This approach offers several advantages and disadvantages:

- The network-based service possesses the resources to employ multiple scanning techniques, such as multiple AV engines, simultaneously to improve detection rates.
- End-user's computers benefit from the replacement of highly-complex AV software with a lightweight client which requires significantly less resources. Furthermore, the less complex client decreases the chance that it could contain exploitable vulnerabilities. This client provides the user with information regarding scan progress and results.
- As the signature database only needs to reside in one centralized location, clients no longer need to continually download updates for their local signature databases. This alleviates problems with bandwidth usage and consumers being out-of-date [OCJ07].
- One disadvantage is that all all clients in the enterprise will try to send the same executable to the cloud. This is avoided in some systems by hashing the file and validating the hash before sending the entire executable into the cloud.

2.4.1.1 Panda Cloud Antivirus. In 2006, Panda Security launched a cloud-based security platform which they call "Collective Intelligence" [Ila09]. The

Collective Intelligence collects and analyzes malware samples, as well as provides various protection mechanisms from the cloud. Panda Security offers several clients that communicate with the Collective Intelligence, including a free, simple AV client for end users as well as more robust network security solutions for businesses.

The three pillars of the Collective Intelligence are:

1. collecting data from the community,
2. automating the processing of that data, and
3. releasing the knowledge extracted back to the community.

Collective Intelligence is able to collect data extremely quickly and effectively due to the cloud-client architecture. Whenever a suspicious executable is detected on *any* client, the file or parts of the file are sent to the Collective Intelligence for further analysis. Panda Security argues that a fundamental problem with modern AV solutions is that malware analysis labs are overwhelmed by the thousands of new malware samples appearing each day [Res07]. The Collective Intelligence automates this process, detecting and classifying suspicious executables without any human interaction. This is achieved through a combination of signature detection, behavioral analysis and blocking, heuristics, as well as new technology for which Panda Security does not provide details. Lastly, Panda Security releases this information through a client interaction with the Collective Intelligence. Panda security offers multiple clients with different capabilities, but the functionality of the basic AV client is the most relevant to this research.

Basically, this lightweight client captures executables before they run on a system and computer a hash of the file. The hash is sent to the Collective Intelligence to determine if it matches existing malware or known goodware. If it is malicious, the file is blocked from executing, if not, the client possesses other tools (such as heuristics and behavioral analysis) to determine if the file is suspicious. Upon discovering a suspicious

file, the end-user is warned and asked to allow the file to be uploaded to the Collective Intelligence. Here, it is analyzed and categorized within six minutes. Now, when any other user executes the same file, they receive an instant report of its legitimacy. The Collective Intelligence takes the process of malware discovery, analysis, signature generation, and signature file distribution, which took days or weeks to fully complete, and reduces it to under six minutes. Furthermore, the burden on the end-user's computer is significantly reduced as all the complex analysis is done in the cloud [Ila09].

There are several drawbacks to cloud-based AV. Panda AV claims that files are analyzed and categorized in less than six minutes, however this does not include the time it takes to upload the file to the Collective Intelligence. Few users are willing to wait six or more minutes to execute a file, and will often ignore the prompts from the AV client to upload the file. Also, when users are disconnected from the internet, Panda's offline detection mechanisms could miss well known malware that signature based AV products would easily detect. Another key drawback to cloud AV is how it detects malware running in memory. This is something that host-based products do frequently, as many malware samples never write themselves to the disk.

2.5 Cloud Malware Detection Framework

The purpose of this section is to extend the literature review to provide some background for the methodology of experimental design presented in Chapter 3. Research regarding cloud computing is synthesized to ensure a cloud computing environment is accurately modeled. More specifically, it ensures that the experiment results from a very simplified version of a cloud environment may be extrapolated into a true cloud computing environment.

Many aspects of NIST's formal definition of cloud computing described in Section 2.1.1 are superfluous to this research as the primary concern is the performance

evaluation of malware detection methods. Each of NIST's essential characteristics and their relationship to this research are listed below.

- *On-demand Self-service and Rapid Elasticity* These cloud capabilities are not directly related to this research. There are no actual consumers in the experiment, and the files on the fileserver are controlled and static. The exclusion of these characteristics from the emulated cloud environment does not preclude the results experimentation to be extended to a cloud environment.

- *Broad Network Access* This capability is a necessary aspect of this research's experimentation methodology. A simple network interface is used that implements standard networking protocols between the fileserver and the detection mechanism. If the fileserver's resources are not made available over a network, the results cannot be extended to a true cloud computing environment.

- *Resource Pooling* This characteristic of cloud computing addresses the underlying architecture of the cloud environment, and should be transparent to the consumer. Likewise, the underlying architecture should be transparent to the detection mechanism, and should not impact performance. Any physical or virtualized environment hosting the fileserver will provide results that can be accurately extrapolated to a true cloud fileserver environment.

- *Measure Service* Again, there is no true producer-consumer relationship in this experiment. Monitoring service usage is not necessary for this experiment. Excluding this capability does not impact the value of research results in a true cloud architecture.

In summary, a simple cloud environment is constructed for the experiments whose results and analysis are extendable and applicable to a real-world cloud environment. Details of the experimental design are presented in the next chapter.

3 Methodology

This chapter presents the methodology this research uses to evaluate the performance of various detection mechanisms operating in a simplified cloud fileserver environment. Section 3.1 defines the problemthis thesis addresses, and Section 3.2 outlines the goals and hypotheses of the research. Section 3.3 discusses the hih-level approach to accomplishing these goals. Sections 3.4 through 3.9 outline the system boundaries, system services, workload, performance metrics, system parameters, and factors. Section 3.10 discusses the evaluation technique, and finally Section 3.11 explains the experimental design.

3.1 Problem Definition

Traditionally, malware detection mechanisms reside on the end-user's machine where they examine local files to discover malware. However, the migration of both file storage and computational resources into the cloud presents several new challenges to the traditional model. First, the transfer of processing power and resources from the end-user into the Internet may render the host machine incapable or inefficient at local detection of malware. Likewise, users' files can be distributed across multiple remote locations. When a host wants to scan its files, it must gather file information from the Internet with likely impacts from available bandwidth. This leaves two alternatives. First, a virus detection mechanism may be employed on each host acting as a fileserver, which could be difficult based on the number of hosts as well as the level of access granted to those hosts. However, with regard to experimentation, this alternative is trivial as it is basically a traditional malware detection model. The other option is to implement a virus detection mechanism in a centralized location, and scan the fileservers remotely. This approach will likely be affected by bandwidth constraints. Little research currently exists to examine how detection mechanisms operate when employed in a network environment.

3.2 Goals and Hypotheses

The purpose of this research is to evaluate the performance of various detection mechanisms employed in a simplified cloud environment. This can be broken down into three goals:

1. Model a simplified cloud fileserver environment in which experiments are conducted whose results are valuable to a true cloud fileserver environment. The network congestion levels within this environment are easily controlled.

2. Evaluate how MaTR performs in regards to file response time when compared to the performance of commercial detection mechanisms when implemented as remote scanners in the modeled cloud fileserver environment and exposed to various network conditions.

3. Evaluate how MaTR performs in the same environment in regards to detection accuracy of known malware and benign files.

In regards to the second goal, this research hypothesizes that MaTR will not outperform the file response times of the other detection mechanisms. MaTR's detection algorithm has proven to be faster than commercial virus scanners in past experiments, but it is currently a proof-of-concept prototype has not been optimized in any way to include handling network congestion. Although MaTR requires only a small amount of information to determine a file's validity, in its current configuration it still needs the entire file to be read into memory. This necessitates transferring the full file across a potentially congested network. This research hypothesizes that the effect of transferring the entire file across a congested network will marginalize MaTR's performance advantage when compared to commercial detection mechanisms. In regards to the third goal, this research hypothesizes that network congestion will not impact MaTR's detection accuracy, and MaTR will continue to demonstrate leading true positive detection rates.

3.3 Approach

To accomplish the aforementioned goals, a simplified cloud fileserver environment is emulated and the performance of various detection mechanisms in that environment is evaluated. A true cloud fileserver has many different configurations and limitations. This research identifies conditions that could affect the performance of a centralized malware scanner, and replicates those conditions with varying severity to conduct experiments to evaluate performance. The observations from these experiments are then compiled and analyzed with the goal of drawing statistically significant conclusions.

3.4 System Boundaries

The System Under Test (SUT) includes the fileserver, the detection server, network connections, and the detection mechanism located on the detection server. Figure 3.1 shows a white box diagram of the System Under Test including inputs and the output metrics that this research measures. The fileservers are populated with benign files as well as known malware. However, these are not part of the System Under Test but rather workload parameters. The Component Under Test (CUT) is the detection mechanism. While many cloud fileservers migh have distributed implementations, replicating a distributed fileserver is not necessary. The network connections to the fileserver and their congestion level are the primary concern. Creating multiple connections to the data with potentially varying congestion levels needlessly complicates the experiments, and make it impossible to draw any valid statistical conclusions. Therefore, the scope of this research will be limited to experiments implementing one detection server with a single connection to one fileserver. The scope of this research is also limited to malware detection and will not quarantine or remove malware. MaTR relies on the Portable Executable (PE) format, and is ineffective at detecting malicious Adobe Reader files or malicious macros in Microsoft Office Files. Therefore, as this experiment is evaluating the effectiveness of

MaTR against other detection mechanisms, the only files being scanned are executables. These experiments make no attempt to optimize any of the detection mechanisms' performance in a network environment. Settings in the AV products are adjusted to maximize available CPU usage.

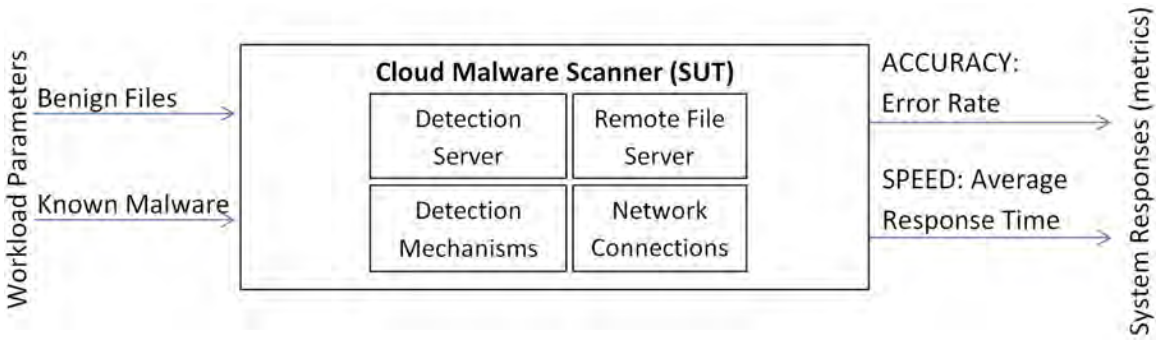

Figure 3.1: Cloud Malware Scanner

3.5 System Services

The SUT provides several services with associated outcomes as a stand-alone system. As the system must store files in remote locations, the SUT provides the service of remote file storage. The outcome of this service is the storage and retrieval of data in a remote location. This implies read/write capabilities, and remote execution is not considered. A potential but rare failure outcome is a hard drive failure and stored files becoming corrupted. Another system service is transportation of data. The outcome of this service is the delivery of data to a remote location. Any transmission failures are assumed to be corrected by the network protocol. Lastly, the system provides various malware detection services. The outcome is the determination whether or not a remote file is benign or malicious. There are several potential failure outcomes when determining file validity. A false positive occurs when a benign file is deemed malicious. False negatives may also occur when malicious files are deemed benign.

3.6 Workload

The only workload submitted to the system are the executables presented to the malware detection mechanism. These include both known malware and benign executable files. Clean files are necessary to populate the fileservers and to accurately model a real-world fileserver. The presence of clean files also allows for evaulation of false positive rates. The clean executables are collected from Windows 2k3 Server Edition and are renamed to their MD5 hash. This creates a unique fingerprint of the file and allows duplicate files to be removed. To maximize the number of unique files, executables are extracted from the baseline installation of Windows Server 2k3, after the installation of each service pack, and after small groups of online updates are installed. For the conclusions of this research to be scientifically valid, experiments conducted in this thesis must not use any of the executables used when developing and training MaTR [DRP+12]. To ensure this, a script is developed that compares executables' hashes gathered for this experiment with the hashes of the executables used in Dube et al's previous work [DRP+12] and deletes the duplicates. This script is found in Section A.1 of the Appendices. After this process is completed, 8,238 unique executables comprise the base of the clean sample set.

Malicious files are obtained from a large online database of known malware called VX Heavens [20111]. This is the same database from which Dube et al. collected known malware specimens in order to train and test MaTR [DRP+12]. As with the clean samples, it is vital that unique samples that were not used to train MaTR are collected in order to gauge MaTR's detection accuracy performance fairly. To accomplish this, an exclusion list was built based on the previous work [DRP+12]. An index of all the malicious filenames hosted on VX Heaven's database is extracted, and 8,270 samples are randomly selected that do not match malware listed in the exclusion list.

MaTR has demonstrated an exceptional capacity in detecting novel malware [DRP+12] [Mer11]. However, novel malware is very difficult to obtain, and as the primary focus of this research is response time performance, novel malware is not included as a workload parameter.

3.7 Performance Metrics

There are two metrics this research uses to evaluate performance. The average amount of time detection mechanisms require to evaluate a file is measured. This is especially important when an "on-demand" cloud detection model is used, or when files are scanned whenever they are accessed. Specifically, this research records average file response time. Response time is the time measured in seconds from when a query regarding a file is sent to the fileserver to when the file's malware classification is ascertained. This information is calculated using

$$Average\ File\ Response\ Time = \frac{Size\ of\ Sample\ Set}{Time\ Elapsed} \tag{3.1}$$

In determining the most effective detection mechanism for a cloud fileserver environment, the ability to accurately identify malware is also essential. Accuracy rates are evaluated for each of the detection mechanisms. This includes true positive and false positive rates, measured as a decimal between zero and one. These rates are easily calculated with

$$TruePositiveRate = \frac{\#Positives_{MS}}{MS}, and \tag{3.2}$$

$$FalsePositiveRate = \frac{\#Positives_{CS}}{CS}, \tag{3.3}$$

where $\#Positives_{MS}$ is the number of malicious file predictions in the malicious sample set, MS is the size of the malicious sample set, $\#Positives_{CS}$ is the number of malicious file predictions in the clean sample set, and CS is the size of the clean sample set.

3.8 System Parameters

System parameters include anything that could affect the performance of the SUT. They are listed below along with explanations of how they could affect the system.

- *Network Congestion* This impacts the rate that data is transferred from the fileserver to the detection mechanism. Network congestion can be influenced through the bandwidth of the connection as well as the frequency of packet loss that the connection experiences.

- *Size of Files* The size of various files being scanned can affect the amount of data that must be sent to the detection mechanism. This can have an impact on the response time demonstrated by the detection mechanisms.

- *Detection Mechanism Used* Several commercial and free malware detection mechanisms are used in the experiment, as well as the novel detection algorithm MaTR.

- *CPU Speed on the Detection Mechanism Host* Insufficient processing resources possessed by the detection server can decrease the performance of the detection mechanisms.

- *CPU Speed of the Fileservers* Insufficient CPU speed on the fileservers can inhibit their ability to respond to data requests.

- *Hard Drive Read Speed of the Fileservers* This affects the amount of time it takes to respond to requests from the detection mechanisms.

- *Network Components Employed* This includes the virtual network adapters, cabling, switches, and routers. Older network protocols and slower components affect the performance of the SUT.

- *Host Operating System (OS)* The host OS affects both the fileserver and the detection server.

3.9 Factors

Factors are system parameters varied to observe their impact on performance. The factors for this experiment are shown in Figure 3.1. The parameters this experiment varies to observe their effect on performance are network congestion and the detection mechanism used. Network congestion affects the bandwidth available to send data across the network. This research tests three levels of network congestion: 1Gbps available bandwidth with no packet loss, 100Mbps available bandwidth with no packet loss, and 100Mbps available bandwidth with 5% packet loss. In all congestion factor levels, there is no additional background traffic other than the basic communication required by the OS and TCP/IP (Transmission Control Protocol/Internet Protocol). Experiments are conducted in a closed network containing only the detection server and the fileserver in order to eliminate extraneous interference contaminating experiment results. The 1Gbps factor emulates the environment of a Local Area Network with maximal connection speeds. The 100Mbps imitates a network environment with more limited connection speeds, such as one might find when communicating across the Internet. The 5% packet loss emulates the effect of a highly congested network with large amounts of background traffic. The level of 5% is chosen based on the results of validation testing with several levels of packet loss. When packet loss is increased much above 5%, file transfers across the network begin failing before they complete. A 5% packet loss demonstrates a significant performance impact on transfer times while consistently completing the transfer process.

The other factor is the detection mechanism used. There are four detection mechanism factor levels implemented in this experiment including three commercial

detection mechanisms and MaTR. This research does not provide the identity of the commercial detection mechanisms.

Table 3.1: Factors

Factor	Network Congestion	Detection Mechanism
Level 1	1Gbps	AV Product A
Level 2	100Mbps	AV Product B
Level 3	100Mbps w/ 5% Packet Loss	AV Product C
Level 4	-	MaTR

3.10 Evaluation Technique

The evaluation technique for this experiment is emulation. An analytic or mathematical evaluation technique is not appropriate for the situation as there is no underlying analytic or mathematical model. The measurement technique is not used because obtaining administrative rights on an actual cloud-based fileserver would be very difficult. Furthermore, it is challenging to control network congestion to observe changes in performance as it would impact the fileserver's operation capabilities. Therefore, this research emulates a simplified version of a cloud based fileserver. The data scanned and the levels of network congestion are controlled and the performance of the detection mechanism is monitored.

As discussed earlier, many cloud fileservers implement a distributed fileserver approach with varying levels of redundancy. Not only is this approach difficult to replicate, but it would make drawing statistical conclusions more difficult and complicate observations. The essential impacts of congestion to a remote malware detection mechanism can be replicated and observed with a single network connection to the data. The experimental setup for this research is illustrated in Figure 3.2. A detection server and

a fileserver are created for each detection mechanism. The fileserver is populated with both malware and clean files. The detection mechanisms remotely scan the malicious and benign files separately, and performance in terms of response time and accuracy rate is recorded. Each detection mechanism is exposed to the same sample sets to ensure the performances are fairly compared. All the servers in the experiments are virtual machines running in VMware Workstation [20110]. Experiments are repeated with varying levels of network congestion introduced. The congestion is modified with built-in congestion controls in VMware. Figure C.1 in Appendix C shows an example of the bandwidth controls available to the LAN segments in a VMware team. Statistical analysis and conclusions are drawn in Chapter 4.

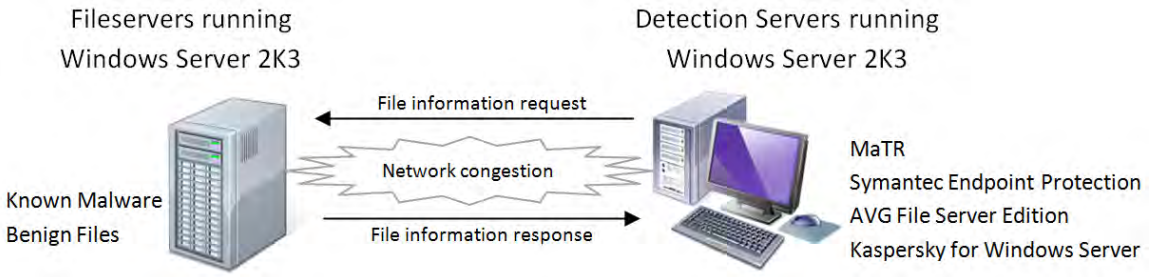

Figure 3.2: Experiment Setup

The components required by the experiment are:

- *Servers* This experiment requires a fileserver and a detection server for each of the four detection mechanisms, a total of 8 servers. These servers are emulated with VMware Workstation 7. Each server is configured with 2GB of RAM, an Intel Xeon X5670 running at 2.93GHz, and a 40GB hard disk drive. The network interfaces are emulated and controlled with VMware Workstation. These hardware specifications should not have a significant impact on the performance of the SUT in comparison to the performance impact of network congestion. These servers are emulated on

two Dell PowerEdge R610 servers. They each have 64GB of memory, two six-core Intel Xeon Processors running at 2.93GHz, and 1TB of storage.

- *Software* Each server runs fully updated Microsoft Windows Server 2003 at the time of the experiment. The fileservers implement Microsoft Sharing Service for simple file and folder sharing to other network devices. VMware Workstation 7 is used to emulate the file and detection servers.

- *Detection Mechanism* The different detection mechanisms installed on the detection server are MaTR and three commercial AV products whose identity is undisclosed. They are referred to as AV Products A, B, and C.

- *Networking Components* The networking components are emulated by VMware Workstation. VMware provides the capability of LAN Segments. When a computer is joined to a segment, a new network connection is created on that machine and it is able to communicate with all other computers joined to the same LAN Segment. VMware allows the bandwidth and amount of packet loss to be controlled within a segment, as shown in Figure C.1 in Appendix C.

There are several ways these experiments are validated to ensure the performance results are legitimate. Preliminary testing with AV Product C shows that there is a significant difference in response time between clean files and malicious files. Malicious files require additional processing and take significantly more time to scan. As such, all mechanisms are exposed to malicious and benign files separately. This is a valuable distinction as various detection mechanisms may respond to the different file types in unique ways. Also, most target environments would not contain the high quantities of malware that this experiment uses to obtain performance observations.

Another validation concern is the performance of VMware's bandwidth controls. To ensure these controls alter bandwidth as expected, a program called LAN Speed Test is

used at each bandwidth factor level. This program measures the transfer rate of data across a network, and substantiates the expected effect of VMware's bandwidth controls.

The volume of experiments that are conducted necessitate that multiple experiments and virtual machines run simultaneously on a single server. Specifically, up to eight virtual machines, and thus four experiments are conducted at a time on the Dell R610 Servers. The simultaneous execution of multiple experiments on a single server could cause a negative impact on performance results of those experiments. To ensure that this is not the case, validation testing is conducted with AV Product C at 1Gbps. Identical experiments are first run consecutively, and then simultaneously. The results of these two validation experiments are compared, and there is no statistical difference in the true mean response time with those experiments. Results of the validation scan are found in Tables B.2 and B.3 in Appendix B. Validation testing is not conducted with any other factor combinations.

The last validation step is ensuring that bandwidth constraints are not impacting detection accuracy of the various mechanisms. This is easily validated by comparing the accuracy results from the different factor levels, and confirming that they are the same.

3.11 Experimental Design

A full-factorial design is used. Each detection mechanism is tested with each network congestion factor level. Equation 3.4 calculates the number of experiments conducted in a full-factorial design.

$$n = \prod_{i=1}^{k} n_i \tag{3.4}$$

A study with k factors with the i^{th} factor having n_i levels requires n experiments. This experiment has two factors. The first factor, network congestion, has three levels, while the second factor, the detection mechanism used, has four levels, resulting in a total of 12 experiments.

To compare the outcomes of these experiments with statistical significance, this research uses 95% confidence intervals. Confidence intervals are used to estimate population parameters (often the population true mean) with a chosen level of confidence given parameters observed in samples (such as the sample mean). For example, response time observations from these experiments are used to calculate a confidence interval for true mean response time that should be understood as, "the *true* mean response time of the detection mechanism on malicious files operating at 100Mbps lies somewhere within the bounds of the confidence interval with 95% confidence". When comparing two detection mechanisms, if the bounds of their confidence intervals overlap *at all*, then there is no statistically significant difference between the mechanisms. If they do not overlap, then there is a statistically significant difference between mechanisms with 95% level of confidence. Confidence intervals are expressed as

$$C.I. = [\bar{x} - ME, \bar{x} + ME] \tag{3.5}$$

where $\bar{x}$ is the observed sample mean and ME is the margin of error. The margin of error is expressed by

$$ME = CV * SE \tag{3.6}$$

where CV is the critical value and SE is the standard error. There are several steps in computing the critical value.

- First, $\alpha = 1 - \frac{confidencelevel}{100}$ is calculated. For all experiments, $\alpha = 1 - \frac{95}{100} = 0.05$.
- Next, the critical probability value is calculated with $p* = 1 - \frac{\alpha}{2} = 1 - \frac{.05}{2} = 0.975$.
- The inverse of the standard normal cumulative distribution with a probability of $p*$ is calculated ("NORMSINV($p*$)" funtion in Excel), which returns the critical value, CV.
- For all experiments, $CV = 1.96$.

Standard error, SE, is calculated with expression $SE = \frac{s}{\sqrt{n}}$ where s is the standard deviation of the sample parameters observed in the experiments, and n is the number of experiments.

In order to use confidence intervals, multiple samples from a population must be collected. This allows multiple sample parameters to be calculated, and a standard error to be determined. To emulate the effect of sampling a population multiple times, this research randomly divides the clean and malicious files into 10 disjoint partitions, each with roughly an equal number of files. For example, 8,270 malware samples are randomly downloaded from VX Heavens, and then randomly divided amongst 10 folders each with 827 files. This is effectively the same as sampling the malware population 10 times, excluding duplicates. Separate scans are conducted on each group of files with the same detection mechanism and the same congestion level. This provides a standard error between the scans, and allows a 95% confidence interval to be calculated for the true mean response time and true mean detection rate for a given detection mechanism at a certain congestion level. This approach also has the effect of replicating each experiment 10 times. An example of an experiment's results and the calculation of the confidence interval are shown in Table B.1 in Appendix B

3.12 Methodology Summary

This chapter outlines the methodology used to create an experiment to evaluate various detection mechanisms in a simplified cloud-based fileserver environment. The research goals are delineated, and the approach to accomplishing those goals is discussed along with hypotheses regarding expected research outcomes. All aspects of the experimental design are fully described, including experimental boundaries, system services, workload, performance metrics, system parameters, and factors. The evaluation technique used is emulation, and details for the construction of the simplified cloud-based fileserver are provided. Performance is measured in terms of average file response time in

seconds and accuracy rates in the form of true positive and false positive rates. The experiment uses confidence intervals to analyze and compare the performance of the detection mechanism.

4 Results and Analysis

This chapter presents and analyzes the results of the experiments outlined in Chapter 3. The first three sections discuss the response times of the scans and are broken down by the congestion factors: 1Gbps, 100Mbps, and 100Mbps with 5% packet loss. Section 4.4 interprets those results, discussing the implication of the data as well as anomalies observed. Section 4.5 analyzes and draws statistical conclusions regarding the detection rates of the various mechanisms. As expected, the congestion factors do not affect the detection rate. Section 4.6 gives an overview of the results and focuses on MaTR's performance compared to the other mechanisms.

4.1 Experiments Conducted in 1 Gbps Environment

This experiment's purpose is to evaluate the performance of the various detection mechanisms with maximal bandwidth available between the detection server and the fileserver. During validation testing of the 1Gbps configuration, the observed transfer rate is substantially lower than the ovailable bandwidth. In other words, the hardware is the limiting factor at this congestion level and does not utilize the full 1Gbps. In this configuration, scan times across the network are only slightly longer than scanning files locally due to the overhead in TCP network protocols.

4.1.1 Malicious Files. Table 4.1 shows the results of the experiment on the malicious files at 1Gbps available bandwidth. Again, 8,270 samples are used which are partitioned into 10 equal groups. These groups provide a measure of variance which is used to calculate confidence intervals, whose bounds are also displayed in Table 4.1. The mean AV Product C response time is larger than the other three detection mechanisms, and the lower bound of the AV Product C response time does not overlap the upper bound of AV Product A, AV Product B, or MaTR. Therefore, it can be said with 95% confidence

that the true mean of the AV Product C response time for malicious files at 1Gbps is greater than the other three detection mechanisms.

Table 4.1: Average Response Time for Malicious Files at 1Gbps in Seconds

Mechanism	Average Response Time	Lower Bound (γ =95%)	Upper Bound (γ=95%)
AV Product A	0.157678356	0.143400729	0.171955982
AV Product B	0.128778718	0.105586773	0.151970663
AV Product C	9.888391778	8.434897885	11.34188567
MaTR	0.033385732	0.030131119	0.036640345

The data from Table 4.1 is shown as a graph in Figure 4.1. The AV Product C data is excluded from the figure as the magnitude of the result is so large it makes comparison of the other mechanisms difficult. The confidence intervals of the AV Product A response time and the AV Product B response time of malicious files at 1Gbps do overlap, so there is no statistical difference between the true mean of their response time. The bounds of MaTR's response time confidence interval not visible due to the relatively small confidence interval width (due to low variance) are not shown in Figure 4.1. As they do not overlap any mechanism, it can be said with 95% confidence that the true mean of MaTR's response time of malicious files at 1Gbps is faster than the other three detection mechanisms.

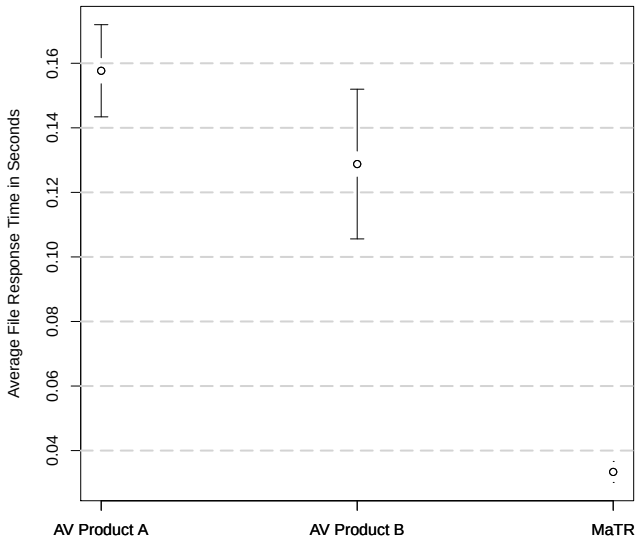

Figure 4.1: Malicious File Response Times at 1Gbps (Excluding AV Product C)

4.1.2 Clean Files. Table 4.2 and Figure 4.2 show the response times of the detection mechanisms when scanning the 8,238 clean files at 1Gbps. AV Product C is still statistically significantly slower than the other three detection mechanisms, but to a lesser degree. The confidence intervals for the response times of AV Product A, AV Product B, and MaTR all overlap, so there is no statistical difference between the true mean response time of clean files at 1Gbps within those mechanisms.

Table 4.2: Average Response Time for Clean Files at 1Gbps in Seconds

Mechanism	Average Response Time	Lower Bound (γ=95%)	Upper Bound (γ=95%)
AV Product A	0.048191556	0.038943646	0.057439466
AV Product B	0.067605345	0.008508202	0.126702488
AV Product C	0.224928335	0.20443024	0.24542643
MaTR	0.04717087	0.032522313	0.061819427

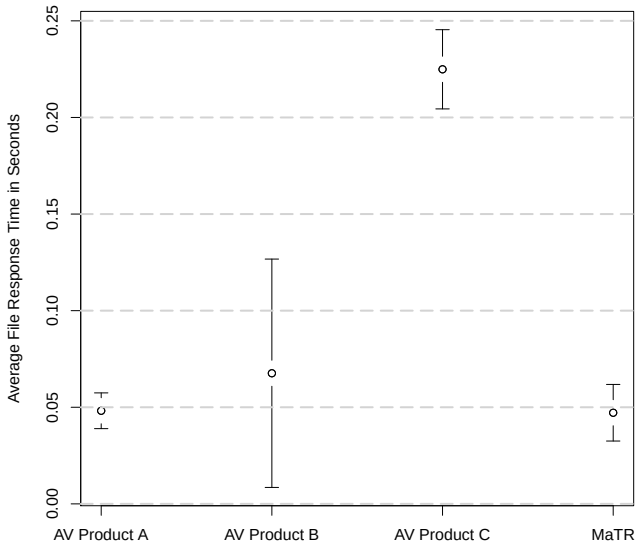

Figure 4.2: Clean File Response Times at 1Gbps

4.2 Experiments Conducted in 100 Mbps Environment

These experiments scan the same files in a network environment limited to a bandwidth of 100Mbps. Validation testing verifies that file transfer rates are limited to the virtual available bandwidth limit and not the hardware. Thus, these results provide an understanding of how the various detection mechanisms perform in regards to file response time when bandwidth is limited. Packet loss is not introduced in these experiments.

4.2.1 Malicious Files. Table 4.3 shows the results of the experiments scanning malicious files at 100Mbps. This data is also graphed in Figure 4.3. Again, AV Product C is significantly slower than the other detection mechanisms to the extent that it makes visual comparison difficult, and thus it is excluded from the figure. While the 95% confidence intervals of AV Product A and AV Product B appear close, inspecting the raw

data in Table 4.3 shows they do not overlap and thus AV Product B has a statistically significant faster mean response time. MaTR proves to have the fastest average response time when scanning malicious files at 100Mbps with a mean of just over 30 milliseconds. The 95% confidence interval for MaTR is so narrow due to low variance that it is not visible in Figure 4.3.

In summary, these experiments demonstrate with 95% confidence that the true mean response time when scanning malicious files at 100Mbps is different for all detection mechanisms. Specifically, the response times follow the following relations

$$MaTR < AV\ Product\ B < AV\ Product\ A < AV\ Product\ C$$

.

Table 4.3: Average Response Time for Malicious Files at 100Mbps in Seconds

Mechanism	Average Response Time	Lower Bound (γ=95%)	Upper Bound (γ=95%)
AV Product A	0.228657799	0.214599315	0.242716284
AV Product B	0.166868198	0.12325677	0.210479626
AV Product C	11.01197098	8.808123583	13.21581838
MaTR	0.031608222	0.030188789	0.033027656

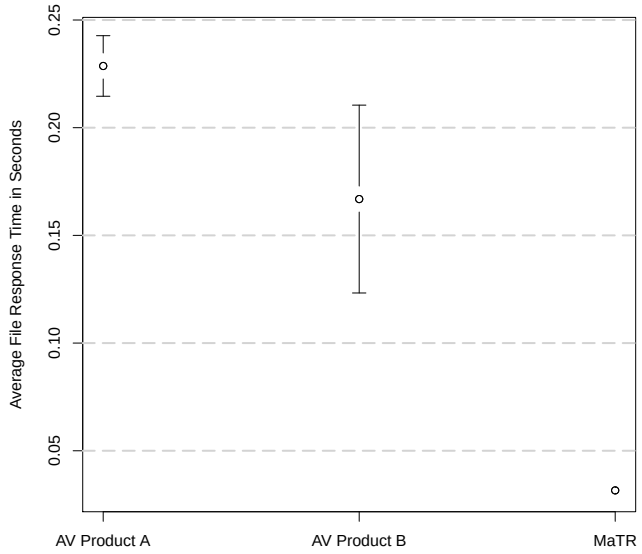

Figure 4.3: Malicious File Response Times at 100Mbps (Excluding AV Product C)

4.2.2 Clean Files. Table 4.4 and Figure 4.4 show the response time results of the experiment conducted on clean files at 100Mbps. Response times are all easily viewed and compared on the same graph. The lower bound of the AV Product C 95% confidence interval is adjacent to the upper bound of the AV Product B confidence interval, but does not overlap, and is well above the other detection mechanisms. Thus this data demonstrates with 95% confidence that the true mean response time of AV Product C while scanning clean files at 100Mbps is slower than the other detection mechanisms. The confidence intervals of AV Product A, AV Product B, and MaTR all overlap, and thus there is no statistically significant difference in their true mean response times.

Table 4.4: Average Response Time for Clean Files at 100Mbps in Seconds

Mechanism	Average Response Time	Lower Bound (γ=95%)	Upper Bound (γ=95%)
AV Product A	0.08909507	0.0600486	0.11814154
AV Product B	0.125861606	0.027765685	0.223957528
AV Product C	0.268512074	0.232256237	0.304767911
MaTR	0.050153963	0.023631253	0.076676672

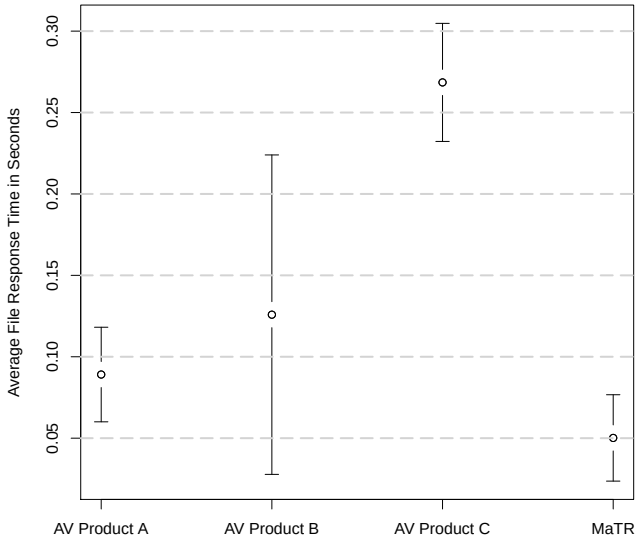

Figure 4.4: Clean File Response Times 100Mbps

4.3 Experiments Conducted in 100 Mbps Environment with 5% Packet Loss

The purpose of these experiments is to evaluate the performance of the various detection mechanisms scanning clean and malicious files in a heavily congested network environment. A bandwidth of 100Mbps is again provided between the detection mechanisms and the fileservers, however 5% packet loss is now introduced. This level of congestion proves to severely impact file transfer rates during validation testing, reducing

it to less than 7Mbps. Results of validation testing also tend to be more inconsistent, as the packet loss induced by VMware is random.

4.3.1 Malicious Files. Table 4.5 and Figure 4.5 show the results of the experiment conducted on malicious files at 100Mbps with 5% packet loss. All the results are statistically significantly slower than the data from previous levels. Again, AV Product C is much slower than the other mechanisms, to the extent that the information is excluded from Figure 4.5. The data demonstrates with 95% confidence that the true mean response time for malicious files is different between each mechanism. Specifically, the response times follow the relations

$$AV\ Product\ B\ <\ AV\ Product\ A\ <\ MaTR\ <\ AV\ Product\ C$$

.

Table 4.5: Average Response Time for Malicious Files at 100Mbps with 5% Packet Loss in Seconds

Mechanism	Average Response Time	Lower Bound (γ=95%)	Upper Bound (γ=95%)
AV Product A	3.259613059	3.113973163	3.405252956
AV Product B	1.526723096	1.325079081	1.72836711
AV Product C	70.96493349	52.37379614	89.55607085
MaTR	8.778113664	7.482280426	10.0739469

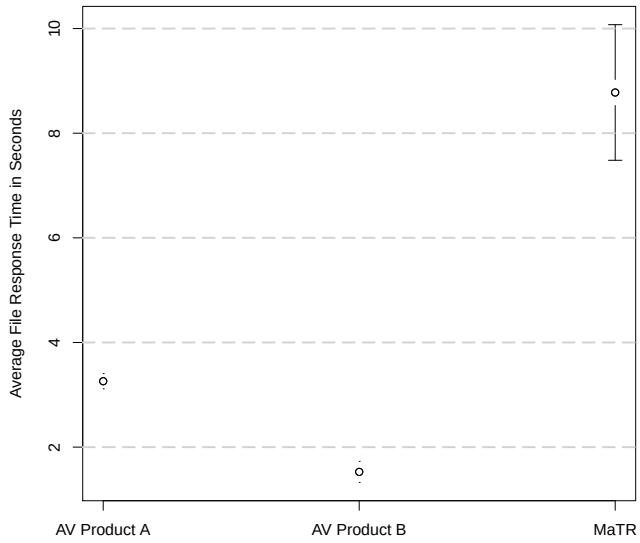

Figure 4.5: Malicious File Response Times 100Mbps with 5% Packet Loss (Excluding AV Product C)

4.3.2 Clean Files. Table 4.6 and Figure 4.6 show the response time results returned from scanning clean files at 100Mbps with 5% packet loss. As with the malicious files, all the response times are slower than the previous congestion levels. When 5% packet loss is induced, MaTR becomes statistically significantly slower than the other three detection mechanisms. AV Product A, AV Product B, and AV Product C's confidence intervals all overlap, thus the data suggests that there is no difference between their true mean response time when scanning with 100Mbps available bandwidth and 5% packet loss.

Table 4.6: Average Response Time for Clean Files at 100Mbps with 5% Packet Loss in Seconds

Mechanism	Average Response Time	Lower Bound (γ=95%)	Upper Bound (γ=95%)
AV Product A	2.566652609	0.740194778	4.393110439
AV Product B	2.312508995	0.801929355	3.823088635
AV Product C	4.191570326	3.349887102	5.03325355
MaTR	13.0294188	6.758817099	19.30002051

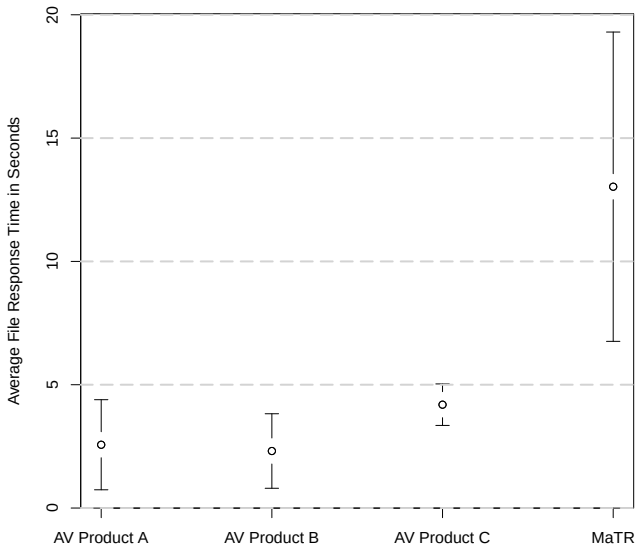

Figure 4.6: Clean File Response Times at 100Mbps with 5% Packet Loss

4.4 Interpretation of Results

This section interprets the data from Sections 4.1-4.3 and synthesizes statistical conclusions about the performance of the various detection mechanisms. This section also highlights unexpected results and anomalies in the data.

4.4.1 100Mbps Congestion Factor's Impact on Performance. When bandwidth is limited from 1Gbps to 100Mbps, validation testing reveals that file transfer rates are significantly impacted. As such file response time for the detection mechanisms is hypothesized to also be impacted as file information must be transferred through the congested network. However, looking at the 95% confidence intervals from the data in Tables 4.1, 4.2, 4.3, and 4.4, this is not the case for AV Product B, AV Product C, and MaTR. The data shows that there is no statistically significant change in true mean response time for each of these three mechanisms when the congestion level changes from 1Gbps to 100Mbps for both clean and malicious files. Figure 4.7 visually demonstrates this for the response time of clean files. AV Product A has a slightly different true mean response time as the 95% confidence intervals do not overlap. However, this difference could be as little as 3ms.

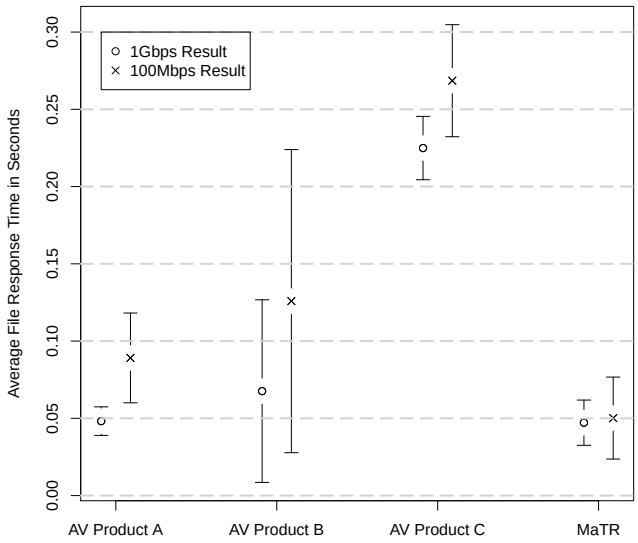

Figure 4.7: Clean File Response Times Across Factors 1Gbps and 100Mbps

The implication of this data is that the AV Product B, AV Product C, and MaTR detection mechanisms do not utilize more than 100Mbps bandwidth when scanning across the network. Instead, the detection algorithms themselves and the computing resources they require are still the limiting factor in the assessment of a file. This observation suggests maximizing available bandwidth to a detection mechanism does not necessarily improve file response time and is wasteful. At some unidentified level of bandwidth, further improvement of performance for each detection mechanism necessitates improving algorithm efficiency, parallelizing the applications, or increasing available computing resources.

4.4.2 100Mbps with 5% Packet Loss Factor's Impact on Performance. Inducing 5% packet loss made a statistically significant impact on the response times of all detection mechanisms. Table 4.7 shows the degree of the change by showing what factor the response times increased by when the 5% packet loss was induced. As the true mean

response time lies within a 95% confidence interval, the table gives both a minimum and maximum impact factor, with the true factor lying between those values. For example, the true mean response time of AV Product A when scanning malicious files slows by at least a factor of ≈12.8 (0.24s to 3.1s) and at most by a factor of ≈15.9 (0.21s to 3.4s). The impact factors are calculated using the bounds of the confidence intervals from 100Mbps response times and the 100Mbps with 5% packet loss response times as shown in the following equations:

$$ImpactFactor_{min} = \frac{LowerBound_{5\%PacketLossC.I.}}{UpperBound_{100MbpsC.I.}} \tag{4.1}$$

$$ImpactFactor_{max} = \frac{UpperBound_{5\%PacketLossC.I.}}{LowerBound_{100MbpsC.I.}} \tag{4.2}$$

The minimum impact factor is measured from the maximum true mean response time (upper bound of confidence interval) of the 100Mbps experiment to the minimum true mean response time (lower bound of confidence interval) of the 100Mbps with 5% packet loss experiment. Likewise, the maximum impact factor is measured from the lower bound of the 100Mbps response time to the upper bound of the 100Mbps with 5% packet loss response time.

Table 4.7: Changes in Response Time Resulting from Inducing 5% Packet Loss

	Malicious Files		Clean Files	
	Minimum Factor	Maximum Factor	Minimum Factor	Maximum Factor
AV Product A	12.82968376	15.86795819	6.2653219	73.15924811
AV Product B	6.295521817	14.0224923	3.580720696	137.6911357
AV Product C	3.962962765	10.16744032	10.99160043	21.67112332
MaTR	226.5459126	333.698276	88.1469805	816.7159116

The data in Table 4.7 should not be used to compare the different detection mechanisms, as they do not directly measure performance. For example, although AV

Product C appears to have a lower impact factor when scanning malicious files, when looking at the data from the experiments the response times are larger than the other mechanisms. While the data in Table 4.7 is not directly useful, it does provide several insights into the performance of the various detection mechanisms.

- Most apparent is that all detection mechanisms are *significantly* impacted by the induction of 5% packet loss, with many of the response times increasing by at least a factor of 10.

- MaTR appears to be impacted the most by the induction of 5% packet loss, with impact factors ranging from 88 to 817. Two inferences are made from this. First, MaTR is still in a development stage. No work has been made towards optimizing MaTR for a network environment, it simply relies on the OS protocol. AV Product A and AV Product B are specifically designed for network environments. This results in sub-optimal responses to lost packets when retrieving file information.

  Secondly, MaTR demonstrates some exceptional response times compared to the other mechanisms with 100Mbps available bandwidth. If all detection mechanisms experience similar response time increases when 5% packet loss is introduced, MaTR's impact factor will be larger than mechanisms with slower initial response times. This can account for some of the impact factor growth.

- The intervals for the minimum and maximum impact factors are much larger for clean files than for malicious files. This is tied directly to the respective confidence intervals of the response times, and is discussed in greater depth in Section 4.4.3. The difference is basically due to greater variance in file sizes of clean files.

4.4.3 Contrasting Clean and Malicious File Response Times. Validation testing with AV Product C showed significant differences in response times between benign and malicious files, regardless of file size. For this reason, the two file types are evaluated

separately, as described in greater detail in Section 3.10. The separation of this data is important as in most situations targets of malware scans will not contain high concentrations of malware, if any. So, even though AV Product C operating in a 100Mbps network environment requires an additional 10s to determine a file's validity when compared to the other detection mechanisms.

This phenomenon is not limited to AV Product C; there are several other instances where a detection mechanism's response time for clean files is slower than that of the malicious files. Another observed difference is that the confidence intervals for the response time on clean files are much larger than the ones for malicious files. Although this disparity in the data is not directly related to this research, it is still worth a brief analysis.

Detailed information on exactly how the various detection mechanisms operate and why response times for malicious files is slower than clean files cannot be obtained. However, a likely hypothesis is that the detection mechanisms perform some sort of 'surface level' scan on each file. This quick scan might determine whether a file is benign or suspicious. If labeled as suspicious, a more thorough investigation of the file may ensue, taking additional time. This theory also supports the findings for MaTR. MaTR implements the same algorithm on both clean and malicious files and at each congestion level the confidence intervals for the clean response times and malicious response times overlap; thus there is no statistical difference between MaTR's true mean response times for clean file and malicious files.

Discrepancies in the confidence interval width between clean and malicious files are easily explained by a greater variance in the file size. The clean files obtained from Windows Server 2k3 contain much larger files, including several update files that were larger than 300MB, a large number of files greater than 10MB, as well as many small files that are only a few kilobytes. The largest malicious file was only 18MB, with an average

file size of around 270KB. This variance in clean file size directly influences the width of the response time confidence intervals.

4.5 Detection Accuracy

This section analyzes the detection accuracy rates between the different detection mechanisms including true positive and false positive rates. Each experiment at the various congestion levels employed the same data set of files. As expected, the various congestion levels did not impact detection rates, and thus congestion is not considered in this analysis.

The data in Table 4.8 shows the average detection rate observed amongst the ten partitions, as well as the bounds of a 95% confidence interval. As with the response time data, the true mean detection rate of each mechanism lies within the bounds of the confidence intervals with 95% certainty. Figure 4.8 shows the data in a graph form for easier comparison. AV Product B and MaTR detection rate confidence intervals overlap with each other, but are higher than the other mechanisms. This means that they have a statistically significant higher detection rate than both AV Product A and AV Product C, but that there is no statistically significant difference between the two. AV Product C has a 95% statistically significant higher true mean detection rate than AV Product A.

Table 4.8: True Positive Detection Rates

Mechanism	Average Detection Rate	Lower Bound (γ=95%)	Upper Bound (γ=95%)
AV Product A	0.972793229	0.967856584	0.977729873
AV Product B	0.9904474	0.988256692	0.992638108
AV Product C	0.983434099	0.980541184	0.986327014
MaTR	0.990810157	0.988282191	0.993338124

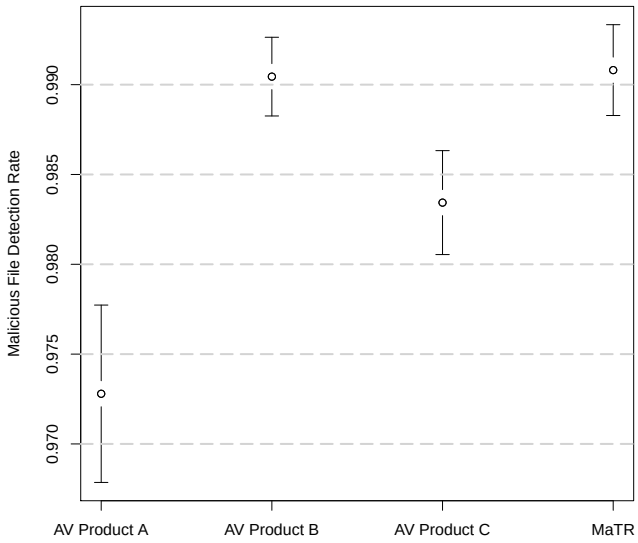

Figure 4.8: Detection Rates for Malicious Files

AV Product A, AV Product B, and AV Product C all have zero false positives when scanning clean files. MaTR, however had 27 false positives out of the 8238 clean files. The false positive rate observed and the true mean's 95% confidence interval is shown in Table 4.9. The data shows that MaTR's false positive rate is less than half a percent.

Table 4.9: False Positive Detection Rates

Mechanism	Average False Positive Rate	Lower Bound (γ=95%)	Upper Bound (γ=95%)
AV Product A	0	0	0
AV Product B	0	0	0
AV Product C	0	0	0
MaTR	0.003277141	0.002210672	0.004343611

4.6 MaTR's Performance and Conclusion

This section focuses on MaTR's performance compared to the other detection mechanisms drawing statistical conclusions and synthesizing explanations for various observations. The data does not reveal a definitive leading performer for all situations. However, MaTR demonstrates a noticeably faster response time than the other detection mechanisms when scanning malicious files at 1Gbps and 100Mbps. This is not the case for clean files or when the congestion level includes 5% packet loss. The following subsections address each of these observations.

4.6.1 MaTR's Response Time Performance at 1Gbps and 100Mbps. As discussed in Section 4.4.3, this research hypothesizes that when some commercial detection mechanisms find a suspicious file, they conduct a more thorough investigation to ensure it is not benign before labeling it malicious. This would explain why observations in the previous sections reveal malicious file response times being slower than benign file response times. MaTR does not contain such a mechanism, and runs the same detection algorithm on every file. This could explain MaTR's faster response times for malicious files as well as having a higher false positive rate than the other mechanisms. When scanning clean files in a 1Gbps and 100Mbps environment, MaTR's true mean response time is not statistically different than AV Product A or AV Product B.

False positives occurring with detection mechanisms can have a critical impact on the target system with certain automatic responses, such as quarantining and deletion. For example, should a false positive occur on an operating system file, it could cause the entire system to crash. MaTR demonstrated 27 false positives, an average 0.33% false positive rate, on executables that are all included in Windows 2k3 Server Edition installation and subsequent updates. Many would consider this unacceptable as these are common operating system files.

In summary, it is difficult to argue that MaTR's overall performance is truly superior to the other mechanisms at 1Gbps and 100Mbps given the false positive rate for clean files and the fact that there is no statistical difference in response time on clean files. As discussed earlier, many commercial target environments generally contain little or no malware so response time for clean files is often more important than for malicious files.

4.6.2 MaTR's Response Time Performance at 100Mbps with 5% Packet Loss. When 5% Packet Loss is introduced into the network environment, MaTR's performance plummets. For malicious files, MaTR is still faster than AV Product C, but statistically slower than AV Product A and AV Product B. For clean files, MaTR is statistically slower than all the detection mechanisms. As discussed in Section 4.4.2, this is likely due to the fact that MaTR is still in development and has not been optimized to operate in a heavily loaded network environment. This version of MaTR should not be deployed in such a network if response times are critically important.

4.6.3 Detection Rates. MaTR demonstrates statistically superior true positive rates than AV Product A and AV Product C. However, there is no statistically significant difference in the true mean detection rate between MaTR and AV Product B; both are exceptional at around 99% true positive rate. MaTR is the only mechanism to exhibit false positives on the benign files.

Although MaTR's performance was not outstanding when compared to the other mechanisms, this experiment did not include novel malware as a sample set. MaTR significantly outperforms commercial detection mechanisms in previous experiments in regards to detecting novel malware [DRP+12] [Mer11]. all capabilties should be kept in mind when considering any AV solution for deployment in a cloud-based environment.

5 Conclusion

This chapter includes a summary of the research conducted and an overview of the observed results. Key conclusions drawn from those results are also discussed. Section 5.2 provides recommendations for future work and potential follow on research.

5.1 Research Results and Conclusions

5.1.1 Goal #1: Construct a simplified cloud fileserver environment. The first goal of this research is to model a simplified fileserver environment in which experiments are conducted whose results can be extrapolated to a true cloud fileserver environment. The network congestion levels are easily controlled for experimentation. This goal is accomplished by creating virtual machines in VMware. A server operating Windows 2k3 is created for the fileserver and the detection mechanism and is connected by a virtual network connection. VMware provides bandwidth and packet loss level controls over this connection. The review of existing literature in Chapter 2 validates that, for the purposes of experiments conducted in this research, experimental observations from this model are valuable in a true cloud fileserver.

Although a true cloud fileserver are likely distributed in nature, the connections to the various servers are the primary factor concerning file response time. Thus, the performance results of the various detection mechanisms when scanning across a single congested connection may be extrapolated to an environment with multiple connections to the same data. Many other aspects of the NIST definition of cloud computing involve capabilities offered to a consumer, and thus are ignored in this research's model as there is no true consumer.

5.1.2 Goal #2: Evaluate MaTR's performance in regards to file response time. The second goal of this research was to observe and analyze the latency of MaTR when

scanning a file in an emulated cloud fileserver and compare that with commercial detection mechanisms. Experiments are conducted at three congestion levels and compared with three other detection mechanisms at the same congestion level. The results of these experiments are summarized by congestion level.

- At 1Gbps, MaTR demonstrates a statistically significant faster true mean response time when scanning malicious files than the other mechanisms. When scanning benign files there is no statistically significant difference in true mean response time between MaTR, AV Product B, and AV Product A. AV Product C is slower with both file types.

- With a 100Mbps congestion level, the comparisons amongst detection mechanisms are identical to 1Gbps. Furthermore, there is no statistical difference in true mean response time of both clean and malicious files when network bandwidth is decreased from 1Gbps to 100Mbps for AV Product B, AV Product C, and MaTR.

- When 5% packet loss is introduced to the 100Mbps congestion level, MaTR's file response time becomes the slowest for benign files, and the second slowest for malicious files outperforming only AV Product C.

5.1.3 Goal #3: Evaluate MaTR's performance in regards to detection accuracy.

The third goal of this experiment is to evaluate how MaTR performs in the same environment with regards to detection rates. These are measured in true positive and false positive rates. As expected, the varying congestion levels do no impact detection accuracy. MaTR demonstrates a true positive detection rate between 98.8% and 99.3% with 95% confidence, which outperformance AV Product C and AV Product A but is not statistically better than AV Product B. However, MaTR exhibits a 0.221% to 0.434% false positive rate, where no other detection mechanisms exhibit false positives.

5.1.4 Research Conclusions. Several key conclusions are drawn from these experimental observations. Reducing the congestion level from 1Gbps to 100Mbps has no impact on the response time performance of AV Product B, AV Product C, and MaTR, as would be expected. AV Product A is slowed by as little as 3ms. The conclusion this research draws is that the unaffected mechanisms do not utilize more than 100Mbps when scanning files across a network. Instead the detection techniques themselves and the computational resources of the detection server are the limiting factor to performance. This is key as network administrators should be aware that increasing bandwidth to a cloud fileserver will not necessarily improve the performance of their detection mechanisms, and other factors should be considered to improve performance.

Another conclusion is drawn from the fact that MaTR is impacted more severely than the other detection mechanisms when 5% packet loss is introduced. The conclusion this research draws is that the commercial detection mechanisms are optimized for network congestion and more robust at handling packet loss, especially AV Product B and AV Product A which are designed for server deployment. MaTR is only a research proof-of-concept prototype currently, and has no optimizations for operating in a network environment. MaTR requires the entire file to be transferred across the network and read into memory using the default protocols of its programming language or the operating system on which it is deployed.

The experimental observations validate that MaTR exhibits excellent detection rates on known malware competing with, and outperforming many commercial detection mechanism. The detection rates and response times observed also suggest that commercial detection mechanisms might take extra precautions against false positives, which requires more time. This accounts for why commercial detection mechanisms have slower true mean response times on malicious files than benign files and lower true positive rates. MaTR performs the same detection technique on all files, which explains

why it outperforms commercial detection mechanisms when scanning malicious files, but has equivalent runtime scan performance when scanning benign files.

The benign files this research uses as a sample set are collected from Windows 2k3 Server Edition and Windows updates. As such, false positives on operating system files could lead to critical failures. MaTR exhibits the only non-zero false positive rates among techniques tested. While the implications to performance are unknown, modifications to MaTR to alleviate its false positives could negatively impact response time rates or reduce true positive rates. As such, although MaTR demonstrates a faster response time on malicious files when provided with sufficient bandwidth, this research concludes that this advantage is somewhat nullified due to the false positive rate.

A final consideration is that this research does not employ novel malware as a sample set. MaTR has demonstrated exceptional performance in detecting novel malware when compared to commercial detection mechanisms in past experiments. Although this capability is not demonstrated in this research, it should be considered when implementing malware detection mechanisms in a true cloud fileserver environment.

5.2 Future Work

This section discusses future work and potential follow on research to this thesis.

- An excellent addition to this research would be to optimize MaTR for a network environment. This could be accomplished in two ways. First, simply optimizing how MaTR handles lost packets. This should allow MaTR to again perform competitively with the commercial malware detectors in terms of runtime performance. However, a more valuable research endeavor would be to change how MaTR collects information from submitted files. Currently, MaTR reads the entire file into memory but needs only a small amount of high-level file information to determine a file's validity. MaTR should be adjusted so that only the information it

needs is requested and transmitted across the network. This could have significant performance improvements when MaTR is deployed in a congested network environment.

- Another future research endeavor would be to make MaTR more robust against false positives, especially when it comes to operating system files. This research could then observe the impact on response time performance, detection rates for known malware, as well as the detection rates for novel malware. These results should be compared with the performance of commercial detection mechanisms.

- An alternative to scanning files remotely would be to create a deployable package implementing the MaTR methodology and sending it to the fileserver, and have it report the results from the remote location. This capability is especially conducive to MaTR as the entire program is a relatively small executable in its current state compared with many commercial detection mechanism's applications.

Appendix A: Scripts

This appendix presents some of the scripts that are used to accomplish the experiments.

A.1 Find and Delete Duplicate Executables

This script is used to find and delete duplicate executables that were used to train and test MaTR in prior experiments [DRP+12]. The script uses the executable's file name (which has been renamed to the hash of the file) and compares it with tthose used in pervious MaTR work [DRP+12].

```
1  Const newFolder = "32" 'folder to filter out already
       existing executables
2  Const existingFolder = "existing" 'text file containing
       existing files
3  Const logFile = "log.txt"
4  Dim fso, log
5
6  On Error Resume Next ' this is set because the designated
       file may be locked from deletion
7
8  ' Instantiate the objects
9  Set fso = CreateObject("Scripting.FileSystemObject")
10 Set eFolder      = fso.GetFolder(existingFolder)
11 set eFiles = eFolder.files
12 Set nFolder = fso.GetFolder(newFolder)
13 Set nFiles = nFolder.files
14 Set eFilesDict = CreateObject("Scripting.Dictionary")
```

```
15
16 ' Create dictionary of existing files (using HASHES)
17 count = 1
18 For Each file in eFiles
19         set currentFile = fso.OpenTextFile(file, 1, True)
20         Do While (currentFile.AtEndOfStream <> True)
21                 endIndex = -1
22                 s=currentFile.ReadLine
23                 For counter=1 To Len(s)-3
24                         if Mid(s, counter, 3)=" ->" Then
25                                 endIndex = counter-1
26                                 exit for
27                         End If
28                 Next
29                 if endIndex <> -1 then
30                         filename = Mid(s, 1, endIndex)
31                         if not eFilesDict.exists(filename)
                              then
32                                 eFilesDict.Add Mid(s, 1,
                                      endIndex), s
33                                 count = count + 1
34                         end if
35                 end if
36         Loop
37 next
38
```

```
39 WScript.echo "Finished building dictionary, " & count &"
     existing files found"
40 'Set up log file
41 Set log = fso.CreateTextFile(logFile, true)
42 log.WriteLine("Duplicate file names that were deleted:")
43
44 'Delete Duplicates
45 count = 0
46 For Each file in nFiles
47         if eFilesDict.Exists(file.name) then
48                 log.WriteLine(eFilesDict.Item(file.name))
49                 file.Delete()
50                 count = count + 1
51         End If
52 Next
53 Wscript.Echo "Total Duplicates Removed: "&count
54 log.WriteBlankLines(2)
55 log.WriteLine("Total Duplicates Removed: "&count)
56 log.Close
57
58 set fso = Nothing
59 set eFilesDict = Nothing
```

Appendix B: Examples of Experiment Results

This chapter contains examples of results from some of the experiments as well as their analysis. Table B.1 is an example of one observation and the calculation of the confidence interval for file respone time. Tables B.2 and B.3 show the results of validation tests to ensure that the execution of simultaneous virtual machines does not impact the performance of the detection mechanisms residing on those virtual machines.

Table B.1: Example of Experiment Results and Confidence Interval Calculations using MaTR on malicious files at 1Gbps

Partition	Time (s)	Files Scanned	Mean File Response Time (s)
1	25.6	827	0.03095526
2	24.7	827	0.029866989
3	21.9	827	0.026481258
4	30.8	827	0.037243047
5	27.7	827	0.033494559
6	23.6	827	0.02853688
7	33.5	827	0.04050786
8	23.6	827	0.02853688
9	31.2	827	0.037726723
10	33.5	827	0.04050786
		Sample Mean	0.033385732
		Standard Error	0.001660547
		Confidence Level	0.95
		Critical Value	1.959963985
		Margin of Error	0.003254613
		C.I. Lower Bound	0.030131119
		C.I. Upper Bound	0.036640345

Table B.2: Validation Test on malware at 1Gbps with AV Product C using Consecutive Scans

Dirty Files (Consecutive Scan)					
Partition #	Start Time	End Time	Time Elapsed (minutes)	# Files	Avg Response Time (seconds)
1	1202	1606	404	827	29.31076179
2	129	521	392	827	28.4401451
3	1404	1754	350	827	25.3929867
4	1925	2330	405	827	29.38331318
5	1311	1721	410	827	29.74607013
6	1527	1933	406	827	29.45586457
7	45	5225	380	827	27.56952842
8	1426	1827	401	827	29.09310762
				Average	28.54897219
				Std Error	0.513382112
				Confidence %	0.95
				Multiplier =	1.959963985
				Lower Bound	27.54276174
				Upper Bound	29.55518264

Table B.3: Validation Test on malware at 1Gbps with AV Product C using Simultaneous Scans

Dirty Files (Simultaneous Scan)					
Partition #	Start Time	End Time	Time Elapsed (minutes)	# Files	Avg Response Time (seconds)
1	1341	1757	416	827	30.18137848
2	1341	1746	405	827	29.38331318
3	1341	1742	401	827	29.09310762
4	1341	1714	373	827	27.06166868
5	1917	2315	398	827	28.87545345
6	1917	2300	383	827	27.78718259
7	1917	2330	413	827	29.9637243
8	1917	2249	332	827	24.08706167
				Average	28.30411125
				Std Error	0.706935763
				Confidence %	0.95
				Multiplier =	1.959963985
				Lower Bound	26.91854261
				Upper Bound	29.68967988

Appendix C: VMware Bandwidth Controls

Figure C.1 shows the bandwidth controls available for the LAN segments in a VMware team.

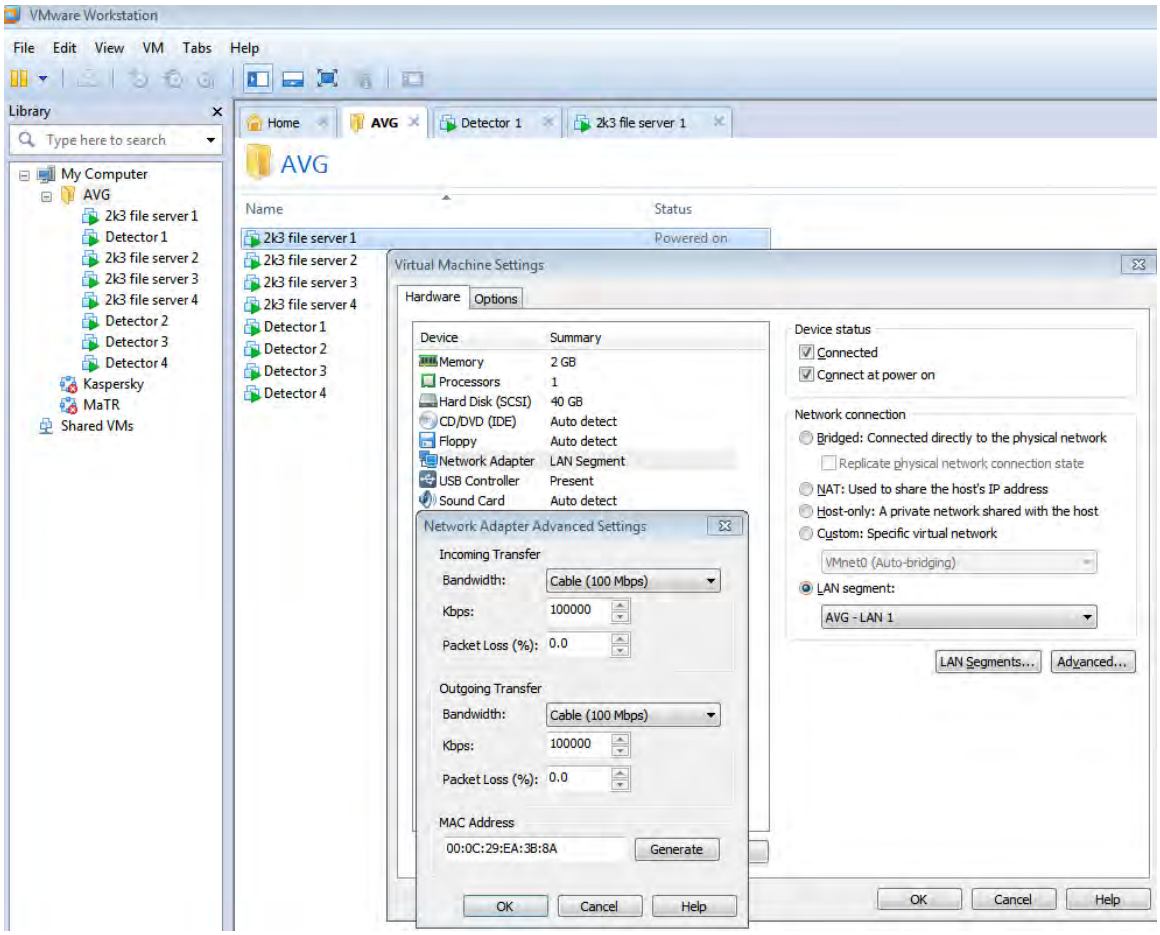

Figure C.1: Bandwidth Controls in VMware

Bibliography

[20110] VMware Workstation 8. Website. http://www.vmware.com/products/workstation/, December 2010.

[20111] VX Heavens. website. http://vx.netlux.org/, February 2011.

[CJ03] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In *Proceedings of the 12th conference on USENIX Security Symposium-volume 12*, pages 12–12. USENIX Association, 2003.

[CJ04] M. Christodorescu and S. Jha. Testing malware detectors. *ACM SIGSOFT Software Engineering Notes*, 29(4):34–44, 2004.

[CJS+05] M. Christodorescu, S. Jha, S.A. Seshia, D. Song, and R.E. Bryant. Semantics-aware malware detection. In *Security and Privacy, 2005 IEEE Symposium on*, pages 32–46. IEEE, 2005.

[Dro11] Dropbox. Dropbox Fact Sheet. Web, http://dl.dropbox.com/u/1656836/presskit/Dropbox%20Fact%20Sheet.pdf, April 2011.

[DRP+12] T. Dube, R. Raines, G. Peterson, K. Bauer, M. Grimaila, and S. Rogers. Malware Target Recognition via Static Heuristics. *Computers & Security*, 31:137–147, 2012.

[Egn11] Egnyte. How it Works - Frequently Asked Questions. Electronic. http://www.egnyte.com/corp/faq.html, April 2011.

[Erd04] G. Erdelyi. Hide'n'seek? Anatomy of stealth malware. In *BlackHat Conference, Amsterdam, Netherlands, Europe*, volume 19, 2004.

[GB11] Benjamin P. Griner and Philip J. Butler. File:Cloud Applications.jpg - Wikipedia, the Free Encyclopedia. Wikimedia Commons, http://en.wikipedia.org/wiki/File:Cloud_applications.jpg, February 2011. Digital image.

[HJ06] O. Henchiri and N. Japkowicz. A feature selection and evaluation scheme for computer virus detection. *Sixth International Conference on Data Mining*, pages 891–895, 2006.

[Ila09] Ionut Ilascu. The Insides of Panda Cloud Antivirus. Electronic, http://news.softpedia.com/news/The-Insides-of-Panda-Cloud-Antivirus-111793.shtml, May 2009.

[IM07] Nwokedi Idika and Aditya P. Mathur. A Survey of Malware Detection Techniques. Technical report, Department of Computer Science, Purdue University, 2007.

[KM04] Jeremy Z. Kolter and Marcus A. Maloof. Learning to detect malicious executables in the wild. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 470–478, New York, NY, USA, 2004. ACM.

[KM06] J.Z. Kolter and M.A. Maloof. Learning to detect and classify malicious executables in the wild. *The Journal of Machine Learning Research*, 7:2721–2744, 2006.

[Mer11] David T. Merritt. Spear Phishing Attack Detection. Master's thesis, Air Force Institute of Technology, 2011.

[MG11] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing. Special Publication 800-145, National Institute of Standards and Technology, September 2011.

[Mic09] Sun Microsystems. Take Your Business To a Higher Level. Electronic, http://www.progression.com/casestudies/studies/Sun_Cloud_Computing.pdf, March 2009.

[OCJ07] Jon Oberheide, Evan Cooke, and Farnam Jahanian. Rethinking antivirus: executable analysis in the network cloud. In *Proceedings of the 2nd USENIX workshop on Hot topics in security*, pages 5:1–5:5, Berkeley, CA, USA, 2007. USENIX Association.

[Res07] Panda Research. From Traditional Antivirus to Collective Intelligence. Electronic, http://www.pandasecurity.com/NR/rdonlyres/08CF7C85-3F3D-4A05-B4AE-3712FA007609/0/02_CollectiveIntelligence_PDF.zip, 2007.

[Ser11] Amazon Web Services. Amazon Elastic Compute Cloud (EC2). Electronic. http://aws.amazon.com/ec2/, March 2011.

[SEZS01] M.G. Schultz, E. Eskin, F. Zadok, and S.J. Stolfo. Data mining methods for detection of new malicious executables. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 38–49. IEEE, 2001.

[SKF08] M. Shafiq, S. Khayam, and M. Farooq. Embedded malware detection using markov n-grams. *Proceeding DIMVA '08 Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 88–107, 2008.

[SWL07] S. Stolfo, K. Wang, and W.J. Li. Towards stealthy malware detection. *Malware Detection*, pages 231–249, 2007.

[SXCM04] A.H. Sung, J. Xu, P. Chavez, and S. Mukkamala. Static analyzer of vicious executables (save). *Proceedings of 20th Annual Computer Security Applications Conference (ACSAC)*, 2004.

[WSSG02] M. Weber, M. Schmid, M. Schatz, and D. Geyer. A toolkit for detecting and analyzing malicious software. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 423–431. IEEE, 2002.

[YA09] Wei Yan and N. Ansari. Why Anti-Virus Products Slow Down Your Machine? In *Proceedings of 18th Internatonal Conference on Computer Communications and Networks, 2009. ICCCN 2009.*, pages 1–6, 2009.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
22-03-2012	**Master's Thesis**	Sep 2010 – Mar 2012

4. TITLE AND SUBTITLE

Evaluation of Malware Target Recognition Deployed in a Cloud-Based Fileserver Environment

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Masters, Gregory P, 2Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCO/ENG/12-08

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States

14. ABSTRACT

Cloud computing, or the migration of computing resources from the end user to remotely managed locations where they can be purchased on-demand, presents several new and unique security challenges. One of these challenges is how to efficiently detect malware amongst files that are possibly spread across multiple locations in the Internet over congested network connections. This research studies how such an environment will impact performance of malware detection. A simplified cloud environment is created where network conditions are fully controlled. This environment includes a fileserver, a detection server, the detection mechanism, and clean and malicious file sample sets. The performance of a novel malware detection algorithm called Malware Target Recognition (MaTR) is evaluated and compared with several commercial detection mechanisms at various levels of congestion. This research evaluates performance in terms of file response time and detection accuracy rates. Results show there is no statistically significant difference in MaTR's true mean response time when scanning clean files with low to moderate levels of congestion compared to the leading commercial response times with a 95% confidence level. MaTR demonstrates a slightly faster response time, by roughly 0.1s to 0.2s, at detecting malware than the leading commercial mechanisms' response time at these congestion levels, but MaTR is also the only device that exhibits false positives with a 0.3% false positive rate. When exposed to high levels of congestion, MaTR's response time is impacted by a factor of 88 to 817 for clean files and 227 to 334 for malicious files, losing its performance competitiveness with other leading detection mechanisms. MaTR's true positive detection rates are extremely competitive at 99.1%.

15. SUBJECT TERMS

Cloud computing, malware analysis, malware detection, fileserver security

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
REPORT	**ABSTRACT**	**c. THIS PAGE**			Dr. Barry E. Mullins, ENG
U	U	U	UU	83	**19b. TELEPHONE NUMBER** *(Include area code)* (937)255-3636 x7979; Barry.Mullins@afit.edu

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18