CREATING NETWORK ATTACK PRIORITY LISTS BY ANALYZING EMAIL TRAFFIC WITH PREDEFINED PROFILES

THESIS

Eric J. Merritt

AFIT/GCO/ENG/12-19

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCO/ENG/12-19

CREATING NETWORK ATTACK PRIORITY LISTS BY ANALYZING EMAIL TRAFFIC WITH PREDEFINED PROFILES

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Eric J. Merritt, BS

September 2012

AFIT/GCO/ENG/12-19

CREATING NETWORK ATTACK PRIORITY LISTS BY ANALYZING EMAIL TRAFFIC WITH PREDEFINED PROFILES

Eric J. Merritt, BS

Approved:

Barry Mullins

Barry E. Mullins, PhD (Chairman)

5 Sep 12

Date

[illegible]

Maj Jonathan W. Butts, PhD (Member)

30 Aug 2012

Date

Thomas E Dube

Maj Thomas E. Dube, PhD (Member)

5 SEP 12

Date

AFIT/GCO/ENG/12-19

Abstract

Networks can be vast and complicated entities consisting of both servers and workstations that contain information sought by attackers. Searching for specific data in a large network can be a time consuming process. Vast amounts of data either passes through or is stored by various servers on the network. However, intermediate work products are often kept solely on workstations. Potential high value targets can be passively identified by comparing user email traffic against predefined profiles. This method provides a potentially smaller footprint on target systems, less human interaction, and increased efficiency of attackers. Collecting user email traffic and comparing each word in an email to a predefined profile, or a list of key words of interest to the attacker, can provide a prioritized list of systems containing the most relevant information.

This research uses two experiments. The functionality experiment uses randomly generated emails and profiles, demonstrating MAPS (Merritt's Adaptive Profiling System) ability to accurately identify matches. The utility experiment uses the Enron email corpus and meaningful profiles generated by onelook.com. This experiment further demonstrating MAPS ability to accurately identify matches with non-random input. A meaningful profile is a list of words bearing a semantic relationship to a topic of interest to the attacker.

Results for the functionality experiment show MAPS can parse randomly generated emails and identify matches with an accuracy of 99 percent or above. The utility experiment using an email corpus with meaningful profiles, show slightly lower accuracies of 95 percent or above. Based upon the match results, network attack priority lists are generated. A network attack priority list is an ordered list of systems, where the potentially highest value systems exhibit the greatest fit to the profile. An attacker then uses the list when searching for target information on the network to prioritize the systems most likely to contain useful data.

Acknowledgments

I would like to express my sincere appreciation for the many people who contributed to this research. Thanks to John Hagen, for not getting upset at my constant interruptions of his second thesis; Cyber Ryan Merritt, for answering my unending questions throughout the process; my family for their constant encouragement to "just keep at it" and to both Major Dube and Major Butts for their inspiration in conceiving this research concept. Finally, thanks to my many friends who completed their thesis before mine, that was the biggest inspiration of all.

Eric J. Merritt

Table of Contents

List of Figures

Figure Page

List of Tables

1. Introduction

1.1 Research Motivation

In a typical computer attack scenario, entry is gained to a network via a node accessible from the Internet. After gaining access, the attacker scans through the network looking for the target information. Attackers search for clues that provide them with the location of the data that they seek, such as systems hosting databases, web servers, and domain controller services. These systems often contain information the attacker is interested in harvesting. The intruder may also attempt to remain in place on a compromised server, capturing data being transmitted across the network such as unencrypted emails, web traffic, and other types of network traffic. Despite the information gathering capability of an attacker's presence on a compromised server, workstations can also provide valuable information to attackers.

1.1.1 Attacker Limitations

Attackers face several problems by limiting their presence to servers. These limitations include: (1) information the attacker seeks may be encrypted on a workstation; (2) documents users create can be stored exclusively on their workstation; and (3) web traffic is encrypted between the workstation and the destination server, which prevents anyone with a presence on the server from intercepting the communications. Without knowing what information is contained by which system, the attacker must gain access to all systems on the network.

1.1.2 Networks

In many enterprise networks, the majority of the workstations applications are similarly configured and cloned from a single configured copy of an operating system. Therefore, profiling a specific workstation and the identity of the user of the workstation is a difficult task for attackers. Compounding the difficulty of finding the target containing the information being sought, it is paramount that the attacker is able to remain stealthy while performing reconnaissance [ScW00].

1.1.2.1 Workstations

By establishing a presence on workstations throughout the network the attacker is able to harvest information such as user credentials, encrypted email traffic, web browsing data, and local documents that are currently being accessed or edited by a user. Gaining access to each system on the network and searching each system for the target information can be extremely time consuming and labor intensive process. Generating a priority listing of the systems most likely to contain the target information increases the efficiency of the attackers while lowering the potential of detection by the network defenders.

1.2 Research Objectives

The goal of this research is to investigate passive methods of identifying potential high value targets by comparing user email traffic against predefined profiles. The following steps are derived in pursuit of this goal:

1. Collect email traffic on a windows workstation and compare the emails to a predefined profile sending the matches to the C2 server.
2. Verify the ability to successfully identify and report matches from a random profile with a random set of emails.
3. Verify the research goal by loading semantically-related profiles and utilizing the Enron email corpus [Enr03] to determine if a network attack priority listing can be derived.

This method provides a potentially smaller footprint on target systems, less human interaction, and increased efficiency of attackers.

1.3 Approach

The research goal is accomplished by monitoring for emails being sent from a notional email application. The text of the emails are compared against a pre-determined profile and matches are recorded along with the sender's email address by MAPS (Merritt's Adaptive Profiling System). These matches are sent to the C2 server, which adds the matches to a database. The attacker can then query the database to obtain the email address, internal and external IP address, the MAC address of each system, as well as the matches they contain. A random profile and random emails are generated from a word list and parsed by MAPS to verify its ability to detect matches. Finally, the email corpus is sanitized, removing extraneous data from the emails. This research focuses on the email author's address and the body of the message. MAPS parses the corpus and compares the text to profiles generated by a dictionary website to verify MAPS abilities to identify matches with semantically related profiles and an email corpus. From these matches, network attack priority listings can be derived. The resulting prioritized list informs the attacker concerning which victim systems appear to contain the most relevant information.

1.4 Research Assumptions

This research accepts several assumptions in order to accomplish the research goal. These assumptions are as follows:

- The attacker has remote access to victim systems without the knowledge of the user using a form of undetected malware.
- Each system on the target network is able to send data to the C2 server using Transmission Control Protocol (TCP)/Internet Protocol (IP).
- Any and all applications are installed necessary to the execution of MAPS.
- The English language is the only language that MAPS is capable of parsing.
- All emails are written using ASCII characters only, and no punctuation is used in the emails with the exception of the @ sign in email addresses. The Enron email corpus is sanitized of all punctuation in the message bodies.

1.5 Thesis Overview

The remaining structure of this document is as follows. Chapter two contains background information including a discussion of profiling techniques, a standard attack model, and a discussion of state-of-the-art methods for finding and retrieving data on a network. Chapter three contains the detailed methodology utilized to implement the various applications, which intercept, analyze, and transmit the results to the C2 server. The methodology also includes details on the experimental setup used to validate the research concept. Chapter four discusses the validation process for MAPS as well as the experimental results. Finally, Chapter five summarizes the research, the methodology, suggested future work, and the results making some final conclusions about the research.

2. Literature Review

2.1 Overview

This chapter provides background and overviews of topics related to this research. Section 2.2 provides three methodologies for creating profiles describing users. Section 2.3 defines a network attack priority list for this research, and Section 2.4 describes an overview of the standard attack model. Section 2.5 presents methods for finding information on a system. Section 2.6 describes techniques for harvesting data from applications. Section 2.7 presents several means of exfiltrating information from compromised systems, and Section 2.8 presents a summary of the chapter.

2.2 Profiling

Considerable research for modeling user behavior has been done. These models are based on various inputs such as server access logs, caches from applications, and browsing histories. Profiling users is a difficult task. A user has the ability to maintain multiple priorities and work flows simultaneously, which can change at any moment. Several research efforts have had success in identifying and predicting a user's preferences and actions. Applications for user profiling ranges from tailoring advertising campaigns and personalizing shopping experiences to detecting rogue users and malicious activity [RBC07].

2.2.1 Profiling Overview

In order to analyze a flow of information, user profiling methods predefine actions, such as browsing to a web page, creating a file, or requesting resources. These definitions vary based on the user profiling methodology. The analysis of actions are incorporated into a profile of the users and accumulating more actions increases the profiles accuracy.

Three methods for accomplishing this are presented in order to give a broad overview of current methodologies. These three methods present varying methods of predefining information for profiling.

2.2.2 User Profiling Methods

2.2.2.1 Activity Ontology

The first method purports that the modeling of an individual user's behavior is performed by recording and analyzing their activities in a browser [Rob10]. This information allows for the tracking of a user no matter where they are on a network via their online behavior. Accomplishing this happens in two phases. The first phase includes the construction of what Dr. Robinson refers to as an "activity ontology" to describe in tree form the performance of any particular activity. An activity ontology contains top level nodes representing a breadth of topics from games and sports to world and religion. The top level nodes contain subcategory layers describing the breadth of the topic. Each of these nodes contains associations which relate that node to other activities in the ontology as well as a "see also" category, containing human edited relationships to other nodes. In the second phase, collection of data from the user in the form of both online and offline data occurs. The composition of online data is URLs, search terms, and page content, while offline data contains bookmarks, cookies, history, and text-based files. The performance of the following six step iterative process on the data generates behavioral profiles.

1. Identify meaningful attributes. This involves identifying attributes which describe a particular group or an individual. These attributes can vary widely and can include occupation, hobbies, and interests.
2. Search activity ontology categories. Using the attributes discovered in Step 1, the activity ontology is queried to determine which activities are associated with what attribute that describe an individual or group.
3. Search activity ontology descriptions. Activity descriptions provide associations to other activities in the ontology. These relate to the attributes identified in Step 1 and the activities discovered in Step 2. In this manner, less obvious activities associated with the attributes are discovered.
4. Identify pertinent "See also" categories. Human identified "See also" categories aid in discovering related categories that may not be intuitive, but are contextually related. Steps 2 through 4 require manual consideration for choosing appropriate activities.
5. Instantiate profile. With the activities chosen, a representative user is chosen that meets the profile activities identified in the previous steps.
6. Refine. The profile generated from the previous steps may provide results that are too narrow or too broad for the purposes required. This step involves analysis of the generated profile in order to modify the profile until it sufficiently describes the individual or group.

Once this behavioral fingerprint has been identified, it can be searched for in a constantly changing network in which the user could be connecting via hotspots, using different workstations, or using mobile network connections [Rob10].

2.2.2.2 Sequence Modeling

The sequence modeling profiling method relies on the theory that "sequences play a crucial role in human skill learning and reasoning". An approach called Evolving Agent Behavior Classification based on Distributions of relevant events (EvABCD) concentrates on modeling sequences as a behavioral profile of a user [IAL09]. This research considers user behavior to be sequences of UNIX commands issued to a computer via a command–line interface. EvABCD has two goals:

1. Creating and updating user profiles from the commands the users type in a UNIX shell.
2. Classifying a new sequence of commands into the pre-defined profiles.

With this behavioral profile the researchers are able to use an evolving system approach to continuously update a user's profile. The system evolves by considering past sequences entered by the user as well as adding in new sequences. A library containing different expected behaviors is created and evolves, influenced by changing user behavior. This adaptive method utilizes an incremental learning algorithm and can be used to detect abnormalities in user behavior for purposes of detecting masqueraders [IAL09].

2.2.2.3 Job History

Yet another method creates user profiles by monitoring the job history of each user that accesses a system [OBB06]. Jobs are requests for processing by a user on a computer center. User models are generated from information obtained during system use by using an evolutionary algorithm that evolves a few generations after each job is completed. The system measures requested resources against actual used resources for a job and utilizes these results as benchmarks. A job queue holds all of the jobs sent by the user while the evolutionary algorithm predicts the actual resources needed to complete all the jobs. Once the system has monitored many interactions, the models are better adapted for the particular user, which allows for the scheduler to be more efficient. The user model can be used in place of the requested resources from the user and also to monitor when the usage of that system by the user has changed.

2.2.3 Information Profiles

The techniques discussed in Section 2.2.2 provide an overview of how user profiles can be generated to describe ever-changing interactions between users and computers. These three methods attempt to build a profile of the user via their past and present activity in order to determine any deviation from it. This research is concerned with pre-defining information in order to identify it when it is present in communication. User profiling typically contains methods for pre-defining information for identification purposes. It is conceivable that these techniques can be adapted to profile information being sought.

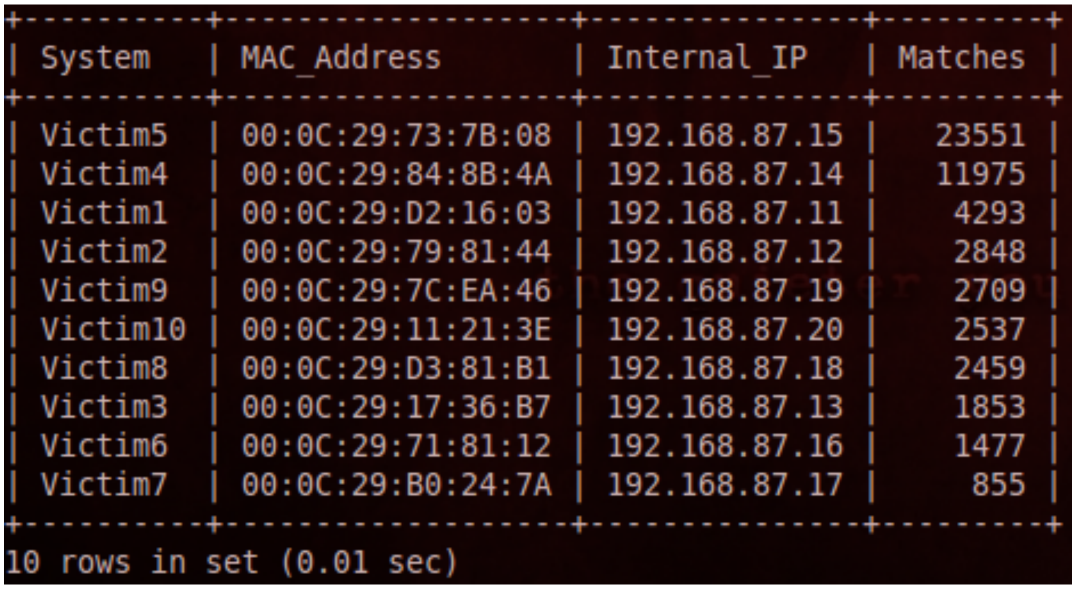
For this research, profiles are a list of words that describe a particular topic. Figure 2.1 shows excerpts from two profiles used in this research. The left profile is a random profile and the right profile is a meaningful profile. Some of the methods described in this section can be used to generate profiles. For this research profiles are generated by two methods: 1) random selection from a wordlist and 2) utilizing a dictionary website with a search word to generate meaningful profiles. A meaningful profile is a list of words that have a semantic relationship to the chosen topic.

goforit	account
john316	activity
sleepy	advertising
claude	affair
iloveyou2	affairs
africa	agency
basil	bank
number9	banking
overkill	brokerage
james1	business
columbia	byplay
randy1	carrier
freedom	clientele
fireball	commerce
scorpio	commercial
gray	company
good	competition
dusty	concern
3010	construction
beanie	corp
micro	corporation
dylan	custom
beaches	deal
power	enterprise
alicia	establishment
monty	firm

Figure 2.1: Random and Meaningful Profile Excerpts

2.3 Network Attack Priority List

A network priority list is an ordered list of systems. The order is determined by the system with the most matches to the profile; the rest of the systems are listed in descending order. Network attack priority listings are useful when looking for specific information on a network. A profile defines the information the attacker seeks. For example, Figure 2.2 presents a sample network attack priority listing with the name, number of matches, and identifying information for each system. The attacker attacks Victim5 first, followed by Victim4, then each subsequent system on the list. This continues until either the information is found, or all systems have been attacked. This list guides the attacker and alleviates the need to guess which system contains the information being sought.



```
+----------+-------------------+----------------+---------+
| System   | MAC_Address       | Internal_IP    | Matches |
+----------+-------------------+----------------+---------+
| Victim5  | 00:0C:29:73:7B:08 | 192.168.87.15  |   23551 |
| Victim4  | 00:0C:29:84:8B:4A | 192.168.87.14  |   11975 |
| Victim1  | 00:0C:29:D2:16:03 | 192.168.87.11  |    4293 |
| Victim2  | 00:0C:29:79:81:44 | 192.168.87.12  |    2848 |
| Victim9  | 00:0C:29:7C:EA:46 | 192.168.87.19  |    2709 |
| Victim10 | 00:0C:29:11:21:3E | 192.168.87.20  |    2537 |
| Victim8  | 00:0C:29:D3:81:B1 | 192.168.87.18  |    2459 |
| Victim3  | 00:0C:29:17:36:B7 | 192.168.87.13  |    1853 |
| Victim6  | 00:0C:29:71:81:12 | 192.168.87.16  |    1477 |
| Victim7  | 00:0C:29:B0:24:7A | 192.168.87.17  |     855 |
+----------+-------------------+----------------+---------+
10 rows in set (0.01 sec)
```

Figure 2.2: Network Attack Priority List

2.4 Attack Model

In the typical advanced attack scenario, an attacker must gain access to a network via some front-facing node accessible from the Internet [SkL06]. In order to infiltrate the network the attacker needs to avoid alerting the network defenders to an ongoing attack. Network defenders are considered to be any human, system, or application that is present on the network that can perform defensive actions when an unauthorized access attempt is made.

2.4.1 Attack Phases

The attacker accomplishes network infiltration by following a standard attack model consisting of five phases [TuS03]. These five phases are shown in Table 2.1, including steps that are included in each of the phases.

Once the attacker has a target in mind, phase one is to discover as much information about the entity that controls that target as they possibly can, gathering lists of systems accessible to the attacker. Phase two is to perform network scans on these targets to identify vulnerabilities that can be exploited to gain a foothold on the network. Phase three exploits the vulnerabilities discovered to gain access to the perimeter systems, which typically include any system that requires access to the Internet including systems such as the DNS name servers, web servers, or any system that provides an interaction between the Internet and intranet of an entity. Once access is gained to the network, phase four is to maintain access to that system by installing some sort of persistent application that will allow the attacker to gain access to the system at any point in the future. Once the first four phases have been accomplished the attacker wants to ensure that they have left the smallest possible detectable presence on the exploited machine. In phase five the attacker cleans up any system access logs and hides any tools that are left on the system for maintaining access. Hiding the tools in place on the system and maintaining access is often accomplished by implanting a rootkit on the system [SkL06]. After the completion of this attack model, the attacker has a pivot point, which allows the attacker to use that system to attack other systems residing on the internal network.

Table 2.1: Anatomy of an Attack [TuS03]

Phase #	Phase Name	Objective	Technique
1	Footprint	Target address range and naming acquisition and information gathering are essential to a "surgical" attack; The key here is not to miss any details.	Search engines, WHOIS database, Web interface to WHOIS, DNS zone transfer
2	Scanning	Target address range, naming acquisition and information gathering are essential to a surgical attack. It is very important not to miss any details.	Ping sweep, Port scan
	Enumeration	Bulk target assessment and identification of listening services focusing on the most promising avenues of entry.	List user accounts, List file shares, Identify applications
3	Gaining Access	Enough data has been gathered at this point to make an informed attempt to access the target.	Password eavesdropping, File share brute forcing, Password file grabbing, Buffer overflows.
	Escalating Privilege	If only user level access was gained in the last step, the attacker will now seek to gain complete control of the system.	Password cracking, Known exploits
4	Acquisition	The information-gathering process begins again to identify mechanisms to gain access to trusted systems.	Evaluate trusts, Search for passwords
5	Cover Tracks	Once total ownership of the target is secured, hiding this fact from the system administrators becomes paramount.	Clearing log files, Hiding tools.
	Back Doors	Trapdoors will be laid in various parts of the system to ensure that privileged access is easily regained at the whim of the intruder	Create rogue user accounts, Schedule batch jobs, Infect startup files, plant remote control services, install monitoring mechanisms, replace apps with trojans

2.4.2 Enterprise Network

Once access to the network is achieved the attacker can now begin the process of locating targets, which can be a daunting task. The United States Air Force (USAF) network contains over a half a million desktops and servers across the world [Lan07]. Enterprise level networks on this scale can be difficult to manage, which has prompted these types of larger enterprise networks to adopt solutions such as the Standard Desktop/Server Configuration (SDC). The SDC is a single image that is vetted and deployed throughout the entire USAF network with common software pre-installed such as an antivirus product, Microsoft Office suite and Adobe Acrobat Reader [Lop06]. This approach has many implications for both the attacker attempting to locate target information across the network and the ability of the network defense team to prevent them.

2.4.2.1 SDC Attacker Advantages

The SDC provides advantages and disadvantages for both the attacker and the defender. Information about the target network is extremely valuable to attackers and the most innocuous release of data can be aggregated with other harmless data. While these pieces of information alone might have been harmless, together they can provide attackers with key information needed to penetrate a secure network. With the SDC an attacker can download the image, explore, and identify potential vulnerabilities that can be exploited to gain access to the system. This has the potential of saving the attacker many hours and giving them a focused plan of attack for penetrating the entire network [MSK09].

2.4.2.2 SDC Defender Advantages

Despite these dangers, by using the SDC the defenders also save many hours and make attacks much more difficult to execute. The SDC images are more secure as a result of only allowing approved commercial software to be present on the system, lowering the privilege level of the average user, and developing the image to secure standards set by various agencies in the security industry including the Department of Defense, National Security Agency, Microsoft, etc. [Lan07].

2.4.2.3 SDC Potential Vulnerability

While this increased security makes it more difficult for the attacker to find and exploit a vulnerability, the entire network becomes more susceptible to zero day exploits that affect software installed on the SDC. Since the configuration is standard throughout a large network, a zero day exploit effective against the SDC can have catastrophic effects, allowing malware to quickly spread throughout the network.

2.5 Finding the Data

Now that the attacker has access to resources on the network, the goal becomes finding the target information. This must be done without alerting the network defenders to the presence of an attacker on the network. There are two types of methods that are available: active methods and passive methods. An attacker is not restricted to using one method or the other, and they will often use a combination of both. Within a network, data is represented as either data-at-rest or data-in-transit. These two types of representation require different methods of interception and retrieval. The following sections delineate active and passive methods of finding targeted data-in-motion and data-in-transit.

2.5.1 Definition of Active Method and Passive Method

Before looking at the methods used to target data, it is necessary to define exactly what is meant by an active method and a passive method. Active methods for finding data include any method that uses stimuli to cause the system to respond in a manner that allows the attacker to gain knowledge of the information contained in or passing through the system. Conversely, passive methods are methods that only observe the flow of data in a system and gain knowledge of information the system contains without applying stimuli to the system that contains it.

2.5.1.1 Active and Passive Network Fingerprinting

Fingerprinting a network with either active or passive methods can be achieved in a similar manner. Figure 2.3 graphically demonstrates the differences between actively and passively fingerprinting a network. When attempting to map a network, a passive network scan can be just as effective as an active network scan with the added benefit of being almost completely invisible to the network administrators [Bar10]. Both active and passive scans can identify open ports and services, map connections and identify operating systems of nodes on the network. However, the increased amount of time required to complete a passive scan is a factor when deciding which scan to use.

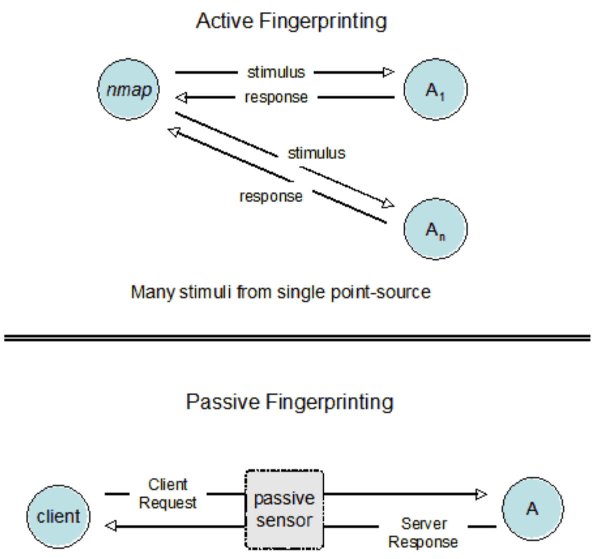


Figure 2.3: Active and Passive Fingerprinting [Bar10]

2.5.1.2 Discovering System Data

There are active and passive methods that can be utilized when attempting to discover data that resides in or passes through a system. The major difference between passive methods of searching data and passive fingerprinting of the network is that the passive data targeting technique requires the attacker to first establish a presence on each system before he can begin monitoring what data flows through that system. These definitions are used in the following sections to categorize the techniques used by attacker to find and exfiltrated data from target networks.

2.5.2 Data-at-Rest

Data-at-rest refers to data that is stored on the computer or an attached storage device [Ide10]. Data can be stored in many different containers including databases, email systems, file shares, and storage area networks [Sha07]. The data stored in these containers can be in many different file formats, ranging from text files to configuration files, as well as a large number of proprietary file types.

2.5.2.1 Active Methods

Active methods of finding data-at-rest often involve the attacker actively searching for the data via a remote terminal. Methods for accomplishing this task include the standard built-in search utilities on the operating system and using domain tools such as DameWare [Dam12].

Most operating systems have a built-in search utility for finding files and folders. The Windows 7 search feature is designed to index external hard drives, networked PC's, libraries, as well as the files on the system itself [Win11]. The operating system uses a program called the Windows Search Service to analyze a document and index information such as the file contents, filename, and file options [Pro11]. While this index does not contain every file, it does index many of the most common file types that the user would commonly search for. An attacker can utilize this index just like any other user on the system via batch scripts designed to search on the command line or manually accessing the search features built into the operating system. Attackers can also make use of other tool sets such as WinSCP [Win12], FAR Manager [Far12], and various other file managers to provide limited scripting and easily available capabilities while accessing a system remotely.

The domain controller is often a primary target for an attacker looking for information and control of a network. The domain controller contains information about users on every system attached to the domain it controls as well as authenticating users to domain resources [Sea00]. Windows domain controllers have a program that is used to manage their domain called the Microsoft Management Console (MMC). However, for an attacker this may not contain all the functionality needed to search the network for the target data. Tools such as DameWare can be installed on the domain controller by the attacker to extend the functionality of the MMC by adding collections of Microsoft administration tools, a remote control program, and powerful export functionality [Ntu11]. With these additional tools and access to the domain controller, the attacker is free to search remotely throughout the entire domain for the target information.

2.5.2.2 Passive Methods

Passive methods for accessing data-at-rest include infecting a system, which either contains the target data, or has access to the target data, with malware. Keyloggers are programs that can be inserted into different areas of the system to log keystrokes that the user enters into the system [HoB06]. Modern keyloggers can also contain the ability to take screen captures of the target system during use to both capture information which is not typed in as well as information entered in via other methods of input [PeI09]. Once the attacker is able to insert the keylogger into the system, the attacker need only wait for the user to access the desired information. Then it is just a matter of the attacker impersonating that user and logging into whatever storage media the data is contained in with the stolen credentials of the user [GBC06].

Other passive means of monitoring what the user accesses on the system falls under a custom Trojan category. Trojan malware hijacks legitimate applications in order to trick the user into allowing access to systems. These custom Trojans can be specifically tailored to the applications that deal with the data being targeted by the attacker. These Trojans lie dormant in the system, waiting for a specific application or file to be launched allowing the attacker access to information on the system.

2.5.3 Data-in-Transit

Data-in-transit can be viewed in several different ways, data moving across untrusted networks such as the Internet and data in transit within an intranet. In addition to this, data-in-transit refers to data that is being processed by the system [Sha07]. Data moving across the Internet is not of interest to this research because this research focuses on activities within the local area networks. In this paper, data that is moving across the intranet and data being processed by the system are both considered to be data-in-transit. This type of data targeting and retrieval can be much more difficult than targeting data-at-rest.

2.5.3.1 Active Methods

Active methods for targeting data-in-transit are some of the most advanced pieces of malware that are currently in use. The design of these specimens requires extensive knowledge of the inner workings and organization of the targeted data, the applications used to encrypt and send the data as well as the operating system that interacts with both. While there are more than just the techniques listed below, the following list represents some of the more dangerous methods attackers can use to intercept data-in-transit:

- Pervasive memory scraping
- Memory parsing malware
- Network injection
- Web form modification

Pervasive memory scraping is one of the most dangerous attack techniques of 2011 according to the SANS institute as well as several other top security threat lists [Mes11]. Many organizations have employed encryption techniques throughout their network to protect data using technologies such as VPNs, SSL, and full disk encryption. These protections protect the data when they are properly employed, but the system must still unencrypt the information in order to be able to process the data [Mil11]. Since the Windows operating system does not always immediately overwrite the memory segments used, the memory scraping malware is able to obtain data that still exists in volatile memory even after the program that was processing that data has terminated [Hel11].

A similar type of active data gathering technique uses malware present on the system to find and dump the memory of applications that are associated with the data that the attacker is searching for. Once this application is identified, the malware dumps the address space associated with that process. A custom memory parsing application can then be employed to dissect the memory dump looking for the target data. This process can be repeated as often as necessary for the malware to intercept the data that is being processed by the targeted application [PSI10]. Attackers can also install debugging tools on the targeted system. This gives them another avenue for both dumping memory spaces to disk and also parsing volatile memory during runtime [Vis08]. Stealthier versions of memory parsing malware inject a memory-parsing module directly into the target process. This eliminates the need for a second process to be running on the system making it more difficult for someone to notice the rogue process. The module can be injected into the process' address space, allowing the malware to search for the target data [VSP10].

An attacker can perform network injection attacks by performing an ARP cache poisoning attack. This technique allows an attacker to remap MAC-IP associations on a network to send packets to the attackers system instead of the intended system such as the gateway. The attacker can then view traffic that was not intended for his system [SkL06] before forwarding it on to the destination without the sender ever knowing the traffic was intercepted. The first benefit to the attacker is that this will allow him to monitor all unencrypted traffic between the client device and any device with which it is trying to communicate. Additional benefits include the ability to modify the data in transit, which allows the attacker to perform a technique called session hijacking or session injection. In a Trustwave penetration test an expert was able to inject their own commands into an existing SQL session between the client and a SQL database [PIM11]. This allowed the attacker the ability to perform tasks such as creating administrative accounts allowing access to the entire database.

An alternate method for gathering data is targeting it at the source. Many corporations, businesses, and other entities utilize a web interface for data entry into databases and various other storage methods. Attackers are able to utilize advanced SQL attacks in order to obtain system-level access to the web interface. With this level of access the attacker is able to modify the contents of the web page to allow for the harvesting of data that is submitted to it [PIM11]. By intercepting the information before it is encrypted and stored, an attacker can receive valuable information without having to break strong data protection schemes by effectively bypassing encryption.

2.5.3.2 Passive Methods

Passive methods for gathering data typically have a much smaller digital footprint on a system once that system has been infiltrated. Since the malware is waiting and watching the data flow without actually affecting it, malware using passive methods can be very difficult to discover. Two methods of passive data gathering are a kernel-level variation of the memory parsing malware discussed above and network traffic sniffers.

The passive version of memory parsing malware operates a bit different than what is described above. Instead of dumping the memory address space of the targeted process, the malware attacks the kernel of the operating system. A kernel-level driver is inserted into the system, which allows the malware to intercept particular system function calls in order to intercept data that is being written to, and read from, the file system. With this access the attacker can search for the target data each time the system either reads from or writes to a file [VSP10].

Sniffing of network traffic is an extremely effective way of discovering sensitive information on both a network and an individual system. An attacker that is able to install a network sniffer on a network and perform an ARP cache poisoning attack will have access to all unencrypted traffic traversing the network. Tools such as Dsniff [Dsn12], Ettercap [Ett12], and Wireshark [Wir12] can be used to listen to protocols on the network and harvest credentials and target information that is traversing the network [SkL06].

Attackers can perform sniffing on the system with keyloggers that are able to intercept the keystrokes entered by the user as they enter data into the system. This information can be used to view user credentials or even be aggregated and parsed in order to find the target data being entered by the user [PIM11].

2.6 Harvesting the Data

Once the target data has been located on the system or the network using one of the techniques above, the next task is to harvest that data in preparation for exfiltration back to the attacker or a command and control server. Which methods are used by the attacker are directly impacted by where the target data is located, what is required to extract the data from memory, and how the data is to be exfiltrated from the target system to the attacker C2 server.

Attackers that utilize one of the memory-parsing malware family techniques can extract the information in several different ways. The attacker can dump the target memory to the disk and then execute a custom parser to look for the target data in the dump. This method has the drawback of using a separate executable to perform each task, which increases the footprint on the target system [PSI10] as well as creating possibly large dump files on the infiltrated computer's hard drive. Other types of targeted malware attacks create their own file on the target system and append any newly found information to the end of the file [VSP10]. Alternatively, some malware never writes any information to disk, instead only keeping the output in volatile memory [PIM11].

2.7 Data Exfiltration Preparation

Persistent attacks often use a file on disk to hold the information the malware is aggregating for the attacker. The attacker has a vested interest in preventing that file from being found prior to the attacker retrieving the information. There are a variety of methods that the attacker can use to make the file unnoticeable, or even if it is found, unreadable. While the attackers may leave the file as an ASCII text file it is much more likely that the attacker will attempt some sort of obfuscation technique to hide their output file [PSI10]. This is accomplished by using various techniques such as encoding, encryption, and steganography.

Encoding is a bitwise operation performing an exclusive or (XOR) between each bit of each character and a secret key, resulting in a "0" or a "1". This operation is run for every bit in the data file resulting in an obfuscated file. This file can only be returned to human readable language by performing the reverse operation with the key [Hus11]. As an anti-forensic capability some malware authors include an encryption algorithm and key with the malware. The malware then encrypts any information written to the output file with the key, requiring anyone wanting to analyze that file to know both the algorithm and key in order to read it [PSI10].

Another more involved method of data hiding is known as steganography. This is the "practice of hiding a message within a larger one in such a way that others cannot discern the presence or contents of the hidden message" [Wes10]. In standard practice this involves hiding data inside a multimedia format such as a picture or music file. This is accomplished by replacing strategic bits of the host file with bits of the data to be hidden [SkL06]. These methods, among others, help the attacker to hide the data on the system while waiting for exfiltration as well as making it easier to bypass perimeter security devices when the actual data exfiltration occurs. This stealthy method comes at a hefty performance price in terms of real data throughput.

While the previous techniques make it difficult for users and automated systems to understand the contents of the attacker's output file, the mere fact of a file's existence increases the chances of someone noticing the penetration of the system. Often attackers turn to file hiding to keep anyone from noticing that the file exists on the system. By default Windows Explorer does not show files that contain the System and Hidden file attributes. By adding these attributes to the output file, the file itself will not be shown to the user if they have not changed the default options for folders [VSP10]. This does not protect the file from being scanned by automated systems but it will prevent a large percentage of users from noticing the file. The location of the output file in the file directory can also aid in preventing users from noticing additional files added by the attacker. While there are no restrictions to where the malware could hide the output file, a couple of the most common file paths are:

- C:\Windows\system32
- C:\Temp
- C:\Documents and Settings*profilename*\Local Settings\Temp [Hus11]

These folders often contain many file types, folders, and executables that are unfamiliar to the average user, making it difficult for them to spot files that should not be there. To further confuse the user, the output file can be changed to an unassociated file type. Since no application has been registered to open that file, anyone attempting to view the contents will be met with an unknown file type error. Since most files in the *system32* folder have the date of the system install, adding a new file to that folder will often stand out from the other files. A method known as time stomping involves changing various fields in the $STANDARD_INFORMATION Attribute in order to modify the creation and modification date shown by Windows Explorer [LAH11]. Malware authors will often mix and match many of these previously discussed techniques to both prevent users from discovering their output files and prevent forensic analysts from discovering the data that was exfiltrated from their networks after the breach has been discovered. The final preparation step of the attacker is often to compress the data that is ready to be exfiltrated thereby reducing the amount of traffic that will be leaving the network. The target data is now ready to be exfiltrated back to the attacker.

At this point the attacker has found the information he sought, extracted it from wherever it resided, written that output to disk, and prevented anyone from finding that output. The last remaining task is to retrieve the target data. Often systems such as web proxies and firewall rules exist at the perimeter of networks in order to control the types of ingress and egress traffic allowed [Coy11]. However there are many ways into and out of a network and it is impossible to secure them all without impeding the legitimate flow of information [GBC06]. The attacker has nearly accomplished his goal and maintaining stealth is still of paramount importance.

2.7.1 Native Protocols

While some malware comes equipped with it's own exfiltration technique, Figure 2.4 shows that the majority of malware uses existing protocols and applications already present on the target system [PIM11].

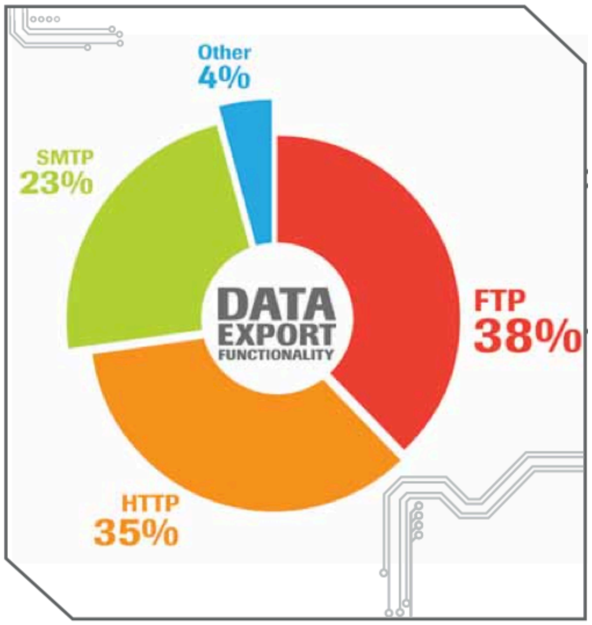


Figure 2.4: Data Export Functionality [PIM11]

There are several applications that exist natively on the Windows operating system that can be used to export data off the system. Some of the protocols that these applications use are the following:

- HTTP (Hyper Text Transfer Protocol)
- FTP (File Transfer Protocol)
- SMTP (Simple Mail Transfer Protocol)

Most networks carry a significant amount of Hypertext Transfer Protocol (HTTP) traffic, making it easier to hide malicious HTTP traffic. It takes a significant amount of processing power to analyze every packet traversing a corporate network, and almost every network allows outbound HTTP traffic [PIM11]. This is also where the data exfiltration preparation has some added benefits. By utilizing one of techniques discussed above it is a difficult task, or even impossible when dealing with encryption, to analyze the traffic that is egressing the network [Gho10] in real-time. In addition many networks do not incorporate Secure Socket Layer (SSL) monitoring so attackers can easily use Hypertext Transfer Protocol Secure (HTTPS) in order to provide built in end-to-end encryption to export their captured data [PIM11]. SSL monitoring works by having a web proxy for SSL communication which hosts a certificate installed on network systems. When a workstation attempts to make an HTTPS connection with a web server the connection is first made to the proxy, which can analyze the traffic. The web proxy will then make a connection to the destination server for legitimate traffic and alert on bad traffic. This allows the web proxy to act as an authorized man-in-the-middle.

FTP is a protocol still in use on many networks, is included in the Microsoft Windows operating system [Per10], and is easy to set up and use to transfer files. These characteristics make it an extremely attractive option for exfiltrating data off the system.

SMTP (Simple Mail Transfer Protocol) is another application often used by attackers to export data [PIM11]. Attackers can either install their own malicious SMTP server or activate the application already installed on Windows operating systems. This allows the attackers to simply email the extracted data to the address of their choosing. While there are many additional ways that attackers can exfiltrate data out of a network, including custom applications, variations on the above applications, and many covert channel techniques, these are often not necessary.

2.8 Summary

This chapter presents overviews and background information on topics related to user profiling, standard attack methodology, active and passive methods for finding information on a system, and exfiltrating data from a network. Attackers are always attempting to find new methods for maintaining access without alerting network defenders to their presence while exfiltrating data out of the network.

3. Methodology

3.1 Introduction

This chapter describes the implementation of MAPS and the various support tools used for validation. From an attacker's perspective, performing reconnaissance on networks is a time consuming and labor intensive operation. Using profiles and automation finding target information on larger networks becomes less difficult. This limits the amount of human interaction with the target network, reducing the chances of detection by network defenders.

3.1.1 Research Motivation

Attackers are often interested in what users discuss electronically via chat and email. This information can be stored on workstations and servers across the network. As the reliance of organizations upon their computer network increases so does the need for the attackers to be able to effectively and easily perform reconnaissance upon those networks. According to the National Public Radio, the U.S. government recognizes that it needs between twenty and thirty thousand computer security specialists in order to both perform operations and defend its networks [Gje10]. Only an estimated one thousand with the required skills currently exist. Motivation for creating tools such as MAPS is the usage of computer security attack experts in a more appropriate manner, allowing tasks that cannot be automated to be assigned to humans.

3.1.2 Chapter Overview

Section 3.2 considers various approaches to accomplishing the research goal. An overview of the implementation is then discussed in Section 3.3, which includes a description of the development environments, tool development decisions, and design decisions. Section 3.4 describes the test environment and the experimental setup in detail, and finally, the summary of the chapter is given in Section 3.5.

3.2 Design Constraints

There are many approaches that can be leveraged to accomplish the goal of determining a ranking of computer systems on a network. This ranking is referred to in this research as a network attack priority list that can be used by attackers to determine the preferred order for finding target information. This research considers the following subset of approaches to be viable options for accomplishing the stated goal:

- Web traffic: Web traffic can provide detailed information about the system's users. However, much more information must be known about each site visited in order to glean any useful information that can be compared to a predefined profile.
- System input logging: The monitoring of input from a user on a system can provide a picture of actions a user performs on a system. However, this type of data gathering would result in a large amount of data that would require complicated monitoring and correlation of input and system access logs to determine user actions.

- Email traffic: This type of traffic is mostly text based and can be easily parsed while providing insight into the system user's communication with other users. In addition, email is a prevalent choice for coordinating group activities. This approach is chosen for the proof of concept.

While many approaches can provide a similar proof of concept, the availability of an email corpus provided a readily available source of input to properly demonstrate the utility of MAPS.

3.3 MAPS Design

This section discusses the implementation of the research tool design known as MAPS, as well as the applications used to support MAPS. This tool intercepts emails from the user in order to determine if the word appears in a profile generated by the attacker. *Startmaps.exe* is placed on the system by the exploitation of a notional vulnerability setting up the folder structure required by MAPS. Once the folders are created, *startmaps.exe* performs a Domain Name System (DNS) query to obtain the Internet Protocol (IP) address of the C2 server and downloads the most recent version of MAPS and the profile. These files are placed in the pre-designated locations in the previously created folder structure.

MAPS is then launched without any windows visible to the user, reading in the profile and waiting for any write operations to occur on the email output file. When a write occurs, MAPS reads the contents of the file, stores the contents in a buffer, and parses the text. The body of the email is compared to the profile, looking for any matches. If no matches are found, then nothing is recorded and MAPS returns to monitoring the file for another message to parse. If matches are found, MAPS records the sender's email address and a list of the matched profile words. Collection of the *to:* field is useful to build associations of users concerned with the topics of interest. These results are recorded in a results file, and MAPS waits for another message to parse.

MAPS asynchronously communicates with the C2 server according to a timer that is configured at compile time by the attacker. When the timer expires, MAPS sends the file containing the matches to the C2 server. MAPS also collects several identifying characteristics of each system it is executing on, including the Media Access Control (MAC) address, IP address assigned by the network Dynamic Host Control Protocol, and the system username. The C2 server parses the message from each MAPS client storing the unique identifying information pertaining to each system in a database.

The attacker is able to analyze the information from a number of clients to build a network attack priority list based upon the matches found. A network attack priority list is an ordered list of systems. The order is determined by the system with the most matches to the profile; the rest of the systems are listed in descending order. An attacker then uses the list when searching for target data on a network.

3.3.1 Development Environments

This research uses three development environments to develop the tools and scripts employed:

- MAPS client development environment
- MAPS server development
- Script development environment

The development system for MAPS client is a machine containing 2 Intel Xeon R2400 quad core processors, 24GB of RAM, 500GB of hard drive space, and running Windows 7 64-bit Enterprise SP1 operating system with the latest updates (at the time of development). The development of MAPS requires the use of a windows compiler; Visual Studio 2010 is used for this purpose.

The second build system is a Virtual Machine (VM) running 1 processor, 8GB of RAM, 20GB of hard drive space, and BackTrack 5 distribution of the Linux operating system. This VM is running on the physical system describe above. The VM setting are chosen as basic settings and have very little impact on the development process. Two utilities are added to the standard BackTrack 5 install:

- Gcc: gcc is the GNU compiler used to compile GNU/Linux executables
- Gdb: gdb is a GNU debugger used to dynamically find and fix programming errors and bugs in the program

The final development environment setup is a system running a 2.53GHz Intel Core i5 processor with 4 cores, 4GB of RAM, 500GB of hard drive space, and Mac OS X version 10.7.4. Only Ruby version 1.9.3 is added to the system in order to begin development of the scripts.

3.3.2 Tool Development

3.3.2.1 MAPS Client

The client portion of MAPS is developed as a 32-bit application targeted to execute on the Windows 7 operating system. While there are myriad operating systems that workstations can potentially employ, as of September 2011 the Windows operating system had a market share of over 86 percent [Net12]. Therefore, it is reasonable to assume that the majority of workstations are operating a version of the Windows operating system. With this in mind, a 32-bit binary is developed in order to provide the greatest cross compatibility between various types of Windows operating systems. Windows 7 is the most current version of Windows at the time of this research.

3.3.2.2. MAPS Server

The server portion of MAPS is developed as a Linux application designed to run on BackTrack 5. BackTrack 5 is a security-oriented Linux distribution that contains a majority of applications pre-installed, decreasing initial setup time and complexity. The presence of a MySQL database setup, python install containing the SimpleHTTPServer script, and the majority of the prerequisites for installing the gcc compiler made this a convenient option as the C2 server for the attacker.

3.3.2.3 Programming Languages

Programming languages utilized throughout development include the C programming language as well as the Ruby scripting language. C is used in the development of the MAPS client and server components as well as the helper application. C is the primary development language due primarily to the author's familiarity, as well as well-documented Windows API calls allowing for difficult tasks to be performed easily. The Ruby scripting language is used in the two scripts that are associated with the email data set. Ruby's choice as a scripting language stemmed from its built in string and regex handling capabilities. Since the scripts are only required to run once, the speed of the program's execution is not a factor, allowing for an interpreted language to be used.

3.3.3 Design Decisions

This section discusses the particular design decisions for this research. These design decisions include:

- MAPS as a multi-threaded application
- MAPS performs asynchronous communication
- MAPS communication method

3.3.3.1 Multi-threaded Application

MAPS has three primary tasks it must perform to accomplish the goal of successfully intercepting user email traffic. First, it must constantly watch for any changes to the file containing the output from the Thunderbird modifications. This file contains the content of the user's email and the address of the user. Secondly, MAPS must analyze a large number of incoming emails simultaneously; finally, MAPS needs to be able to contact the C2 server at intervals set by the attacker. To do this, MAPS makes use of the *process.h* standard C library. Figure 3.1 illustrates the MAPS process flow discussed throughout this section.

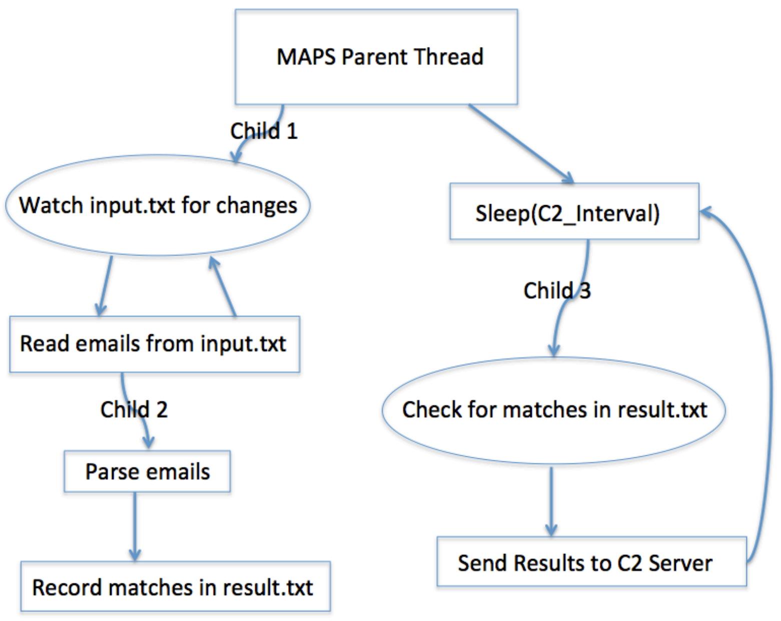

MAPS Parent Thread
Child 1
Watch input.txt for changes
Read emails from input.txt
Child 2
Parse emails
Record matches in result.txt
Sleep(C2_Interval)
Child 3
Check for matches in result.txt
Send Results to C2 Server

Figure 3.1: MAPS Process Flow

In order to accomplish the three tasks listed above, the main function, referred to as the parent thread, creates two threads.

The first thread, referred to as child 1, calls the function `waitForFileWrite`, which expects no arguments and is shown in Figure 3.2.

```
_beginthread(waitForFileWrite, 0, NULL);
```

Figure 3.2: Begin Child 1

The `waitForFileWrite` function utilizes the Windows API call `FindFirstChangeNotification` with three arguments, shown in Figure 3.3. The path of the folder to be watched, a flag indicating if any subfolders within that folder should also be monitored, and a filter specifying which change notification satisfies the function.

```
h = FindFirstChangeNotification( wWatchFolder, 0, FILE_NOTIFY_CHANGE_LAST_WRITE );
```

Figure 3.3: Watch Folder Code

Child 1 tells the operating system to signal when there is any change to the last write-time of files within the watched directory [Msd2]. This function causes child 1 to wait until such an even occurs. Upon continuing, child 1 then opens the file and ingests the contents into a dynamically allocated buffer called `email`. Figure 3.4 shows a new thread, child 2 is spawned in order to parse the contents of `email`. This new thread begins in the function `checkFile`.

```
_beginthread(checkFile, 0, (void *)email);
```

Figure 3.4: Begin Child 2

`checkFile` then performs the task of parsing the text of the email and determining if any matches occur. Child 1 returns to waiting for changes to occur in the watch directory. In this manner, the parsing of the emails contents, which can be time consuming, does not interfere with the MAPS function of waiting for emails to be written to the output file.

Figure 3.5 shows the parent thread creating the second thread, referred to as child 3. Child 3 calls the function callC2, which is expecting no arguments.

```
handle = (HANDLE) _beginthread(callC2, 0, NULL);
```

Figure 3.5: Begin Child 3

This thread call is executed repeatedly within a while loop throughout the execution of MAPS. The loop employs a sleep timer that causes the parent thread to sleep for a time period specified by the attacker during compilation. This timer is defined in the source code shown in Figure 3.6.

```
#define C2_INTERVAL 300000    //phone home interval - 1000/second
```

Figure 3.6: C2 Contact Timer

This interval causes the parent thread to sleep for 300 seconds before spawning the child 2 thread again.

```
while(1)
{
        Sleep(C2_INTERVAL);
        handle = (HANDLE) _beginthread(callC2, 0, NULL);
        WaitForSingleObject(handle,INFINITE);
}
```

Figure 3.7: C2 Send Results Loop

The `WaitForSingleObject` API call, shown in Figure 3.7, takes in the handle to the thread being called and causes the parent thread to wait indefinitely, or until child 2 finishes executing. `callC2` performs the task of reading the matches from *results.txt*, contacting the C2 server, and sending the match information.

3.3.3.2 Asynchronous Communication

The asynchronous communication used in MAPS allows the attacker to control two things. First, the attacker can control when the matches are communicated back to the C2 server. If the attacker has knowledge of the target network, the attacker is able to customize the time of day communication occurs with the C2 server. The attacker can either hide during busy network traffic times or communicate when network defenders will not notice. The second benefit of this asynchronous communication method is the ability of the attacker to configure MAPS network noise level based upon the need to remain unnoticed on the network. Exfiltrating the data with longer intervals makes MAPS less likely to be noticed by a network defense team. Overall, sending transmissions on a modifiable timer back to the C2 server allows the attacker to configure the tool for the target network.

3.3.3.3 C2 Communication Method

When the `C2_INTERVAL` expires, MAPS enters the `callC2` function in order to set up a communication link with the C2 server. Once inside this function, MAPS determines if any matches have been recorded, shown in Figure 3.8. If *results.txt* contains no matches, then the function exits and waits for the `C2_INTERVAL` to expire. This prevents extraneous communication from occurring that could be detected by the network defenders.

```
pFile = fopen ( resultPath , "rb" ); //open the result.txt file
if (pFile==NULL)
       _endthread(); //if the file fails to open, exit thread

fseek (pFile , 0 , SEEK_END); //set position indicator to end of file
lSize = ftell (pFile); //get value of position indicator; gives file size
rewind (pFile); //set position indicator to beginning of file

fclose(pFile); //close results.txt file
if(lSize == 0)
{
       _endthread(); //if the file is empty, exit the thread
}
```

Figure 3.8: Write Results

If MAPS is unable to open the file or if the file is empty, the thread exits. Once MAPS determines that matches need to be exfiltrated, MAPS checks the configuration setting for the hostname of the C2 server shown in Figure 3.9.

```
char* DNSNAME = "thisIsNotMalware.com";
char* C2IP = NULL;
```

Figure 3.9: MAPS Configuration Settings

The C2 server's hostname is `thisIsNotMalware.com`. `callC2` calls another function, `queryC2` and perform a DNS query for the hostname to obtain the IP address of the C2 server. The attacker can change the IP address associated with the hostname from anywhere without making any changes to MAPS. This flexibility allows the attacker to configure MAPS to target the defenses of a network and prevent a single IP address from showing up repeatedly in network communication logs.

3.4 Test Environments and Experimental Design

In order to test MAPS, VMs residing on a virtual network are employed. Virtual machines allow for each test to be run from the same system state, reducing the differences in the system that might occur between tests. Utilizing a virtual network also allows for the experiment to be run without interfering network traffic or network noise from non-pertinent systems on the network.

3.4.1 Database Setup

A MySQL database is employed on the C2 server to record the information sent from the clients. Figure 3.10 shows the schema for the database.

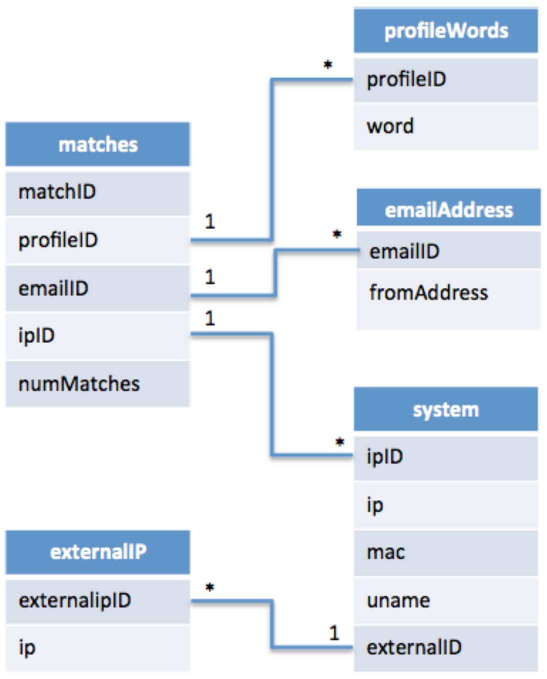


Figure 3.10: C2 MySQL Database Schema

Each message sent from the client to the C2 server contains a MAC address, internal IP address, system username, email username, match words, and the number of occurrences of that match word. The server itself also records the IP address of the incoming connection from the client. First, the server queries the *externalIP* table to establish if this IP address has communicated with the C2 server before, determining if this is a new system. This query is shown in Figure 3.11.

```
sprintf(command, "SELECT externalipID FROM externalIP WHERE ip = '%s';", ip);
if (mysql_query(conn, command)) {
        fprintf(stderr, "%s\n", mysql_error(conn));
        exit(1); }
```

Figure 3.11: MySQL Query for External IP Address

The *externalipID* is returned if a record exists. Otherwise, the new value is inserted into the *externalIP* table and the new *externalipID* is returned. The server then queries the *system* table to determine if the internal IP and external IP combination exists. This is shown in Figure 3.12.

```
sprintf(command, "SELECT ipID FROM system WHERE ip = '%s' AND externalID = %d;",
  input[0], externalID);
if (mysql_query(conn, command)) {
  fprintf(stderr, "%s\n", mysql_error(conn));
  exit(1); }
```

Figure 3.12: MySQL Query for Internal/External IP Combination

If it exists, the *ipID* of that system is returned. Otherwise, a new entry is inserted in the system table, and the new *ipID* is returned. Figure 3.13 shows the server querying the *emailAddress* table to determine if the email address exists in the database.

```
sprintf(command, "SELECT emailID FROM emailAddress WHERE fromAddress = '%s';",
        input[loop]);
if (mysql_query(conn, command)) {
        fprintf(stderr, "%s\n", mysql_error(conn));
        exit(1); }
```

Figure 3.13: MySQL Query for Email Address

The *emailID* is returned if a record is found. Otherwise, a new entry is added to the *emailAddress* table and the new *emailID* is returned. The server loops through each match word sent from the client, querying the *profileWords* table to get the *profileID*'s of each match. This is shown in Figure 3.14.

```
for(i = 0; i < numMatches; i++)
{
        memset(command, '\0', commandMem);
        sprintf(command, "SELECT profileID FROM profileWords WHERE word='%s';",
                matches[i]);
        if (mysql_query(conn, command)) {
        fprintf(stderr, "%s\n", mysql_error(conn));
        exit(1); }
}
```

Figure 3.14: MySQL Query for Profile Word ID's

Figure 3.15 shows the query of the match table with the identifiers to determine if a `matchID` exists for this combination.

```
sprintf(command, "SELECT matchID FROM matches WHERE profileID=%d AND emailID=%d AND
         ipID=%d;", profileID, emailID, internalID);
if (mysql_query(conn, command)) {
         fprintf(stderr, "%s\n", mysql_error(conn));
         exit(1);  }
```

Figure 3.15: MySQL Query for Match ID

If it exists, the *numMatches* count is updated with the new matches. Otherwise, a new `matches` entry is added to the database with the profile word, email address, and the new system identifiers.

3.4.2 Virtual Network Setup

Virtualization takes place on a single host system described in Section 3.3.1. The virtualization software installed is VMware Workstation version 8.0.1. The virtual network is a Host-only private network, which uses the host as the gateway for all nodes in the network. Two operating systems are utilized. The first operating system installed is the BackTrack 5 operating system described in Section 3.3.1. The second operating system which is duplicated depending upon the experimental need, is running Windows 7 64-bit Enterprise SP1 operating system with 1 processor, 1GB of RAM, and 30GB of hard drive space.

3.4.2.1 VM IP addresses

Each operating system is statically assigned an IP address. This prevents IP conflicts from occurring with the DHCP assignment when reverting a VM to a snapshot. Two IP addresses are recorded during the running of the experiments:

- MAPS client records the IP address of the network interface on the system and sends it to the C2 server when matches are reported. The C2 server parses the message storing client IP address in the *internalIP* table of the database.
- During transmission, the C2 server records the IP address of the incoming connection. The IP address is stored in the *externalIP* table of the database.

Discrepancies between these two IP addresses cause a new system identifier to be added to the database instead of adding matches to the currently existing system identifier. Figure 3.16 shows the network diagram for the virtual network. The gateway is a virtual network interface on the host operating system.

Both the external and internal IP addresses are stored in the database allowing the attacker to properly identify a system on a network that is employing Network Address Translation (NAT). NAT permits a network to use private IP address space, allowing multiple computers to use the same Internet facing IP address. With both the internal and external IP addresses, the attacker is able to identify a system behind a router employing NAT.

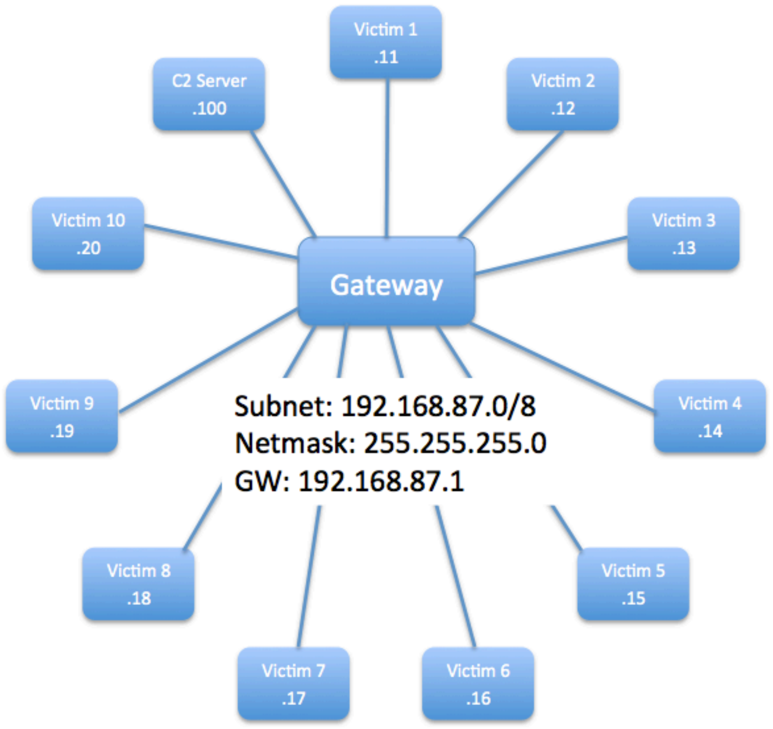


Figure 3.16: Virtual Network Layout

3.4.3 Experimental Overview

The experiments are designed to show both the functionality and utility of the concept to create network attack priority list. A successful proof of concept is one that accurately reports matches that exist in both the email and the profile. In order to validate MAPS, two different experiments are run.

3.4.3.1 Functionality Experiment

The functionality experiment utilizes a word list to randomly generate both emails and profile lists. The purpose of this experiment is to demonstrate the functionality of

MAPS and it's ability to properly collect emails, parse them for content, compare them to the profile, and exfiltrate the matches and system identifiers to the C2 server.

3.4.3.2 Utility Experiment

The utility experiment uses an email corpus while the profiles are generated from the online dictionary site having no predetermined relationship to the email dataset. The purpose of this experiment is to determine MAPS ability to work with non-contrived input.

3.4.3.3 Experiment VMs

Two types of VMs are used for both the functionality and the utility experiment: C2 server and victim machine. The BackTrack 5 operating system is used as the C2 server and Windows 7 64-bit Enterprise SP1 operating systems serves as a victim machine. There is one C2 server and the victim machines are duplicated as required for the experiment (i.e., if there are 3 victims running they are labeled as Victim1, Victim2, and Victim3). ApateDNS [Man12], an application that allows for all DNS queries to return a single IP address, is running on all victim machines and the IP address of the C2 server is returned for all DNS queries. The C2 server is running the python script SimpleHTTPServer hosting *maps.exe* and *profile.txt* for all experiments.

3.4.4 Functionality Experiment

The functionality experiment uses the word list packaged with John the Ripper [Ope12] password cracker, edited to remove inappropriate language, consisting of 3075 words. This list is further referred to as the master profile.

3.4.4.1 C2 Server Setup

Figure 3.17 demonstrates the C2 server setup that is described in this section.

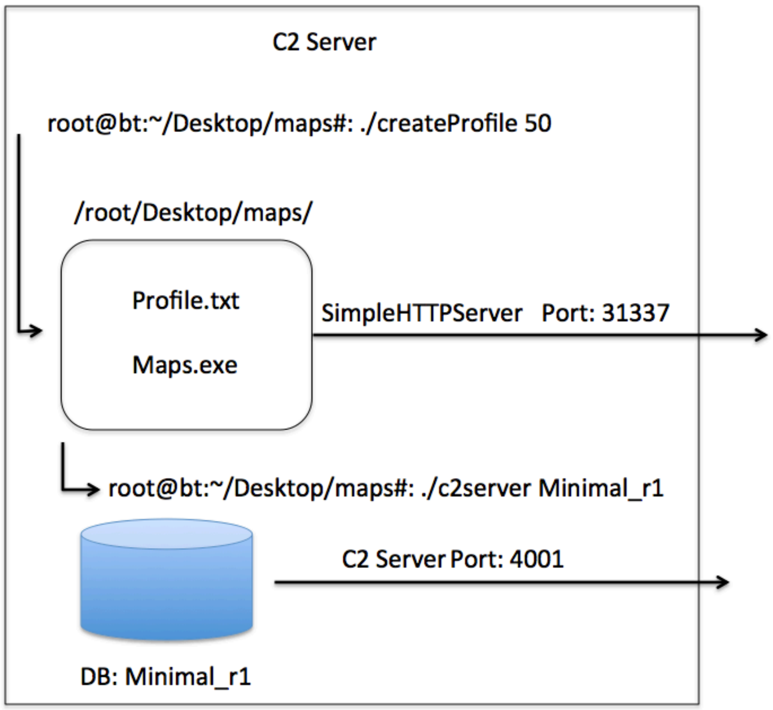

C2 Server
root@bt:~/Desktop/maps#: ./createProfile 50
/root/Desktop/maps/
Profile.txt
Maps.exe
SimpleHTTPServer Port: 31337
root@bt:~/Desktop/maps#: ./c2server Minimal_r1
C2 Server Port: 4001
DB: Minimal_r1

Figure 3.17: C2 Server Setup

The tool *createProfile* is run on the C2 server and accepts, as a command line argument, the number of unique words to use for the profile. The specified number of words is pseudo-randomly selected from the master profile, using a properly seeded srand function from the standard C library to create a profile. Once a profile is generated, the C2 server hosts *maps.exe* and *profile.txt* by executing the SimpleHTTPServer python script on port 31337. The *C2server* program is executed with the name of the database to create, reads in the *profile.txt*, creates the database, and begins listening for connections on port 4001.

3.4.4.2 C2 Client Setup

Figure 3.18 shows the folder structure created when *startMaps.exe* executes.

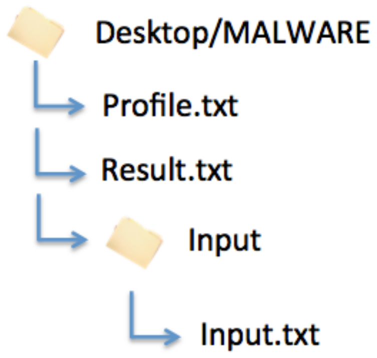


Figure 3.18: MAPS Client Folder Structure

Next, *startMaps.exe* performs a DNS query on the hardcoded hostname thisisnotmalware.com to determine the IP address of C2 server. After successfully receiving an IP address, *maps.exe* and *profile.txt* are downloaded to the appropriate folders on the victim machine. *StartMaps.exe* finishes execution by launching *maps.exe* as a background application with no window interface shown to the user. When MAPS begins its execution, *profile.txt* is read into memory and stored as an array. MAPS then begins monitoring the *input.txt* for emails.

3.4.4.3 Functionality Experiment Data

With MAPS running on Victim1, executing *generateEmails.exe* produces the emails for the experiment. *GenerateEmails.exe* requires three arguments:

1. The number of emails to be generated.
2. The amount of time *generateEmails.exe* should sleep in milliseconds between generation of emails.
3. The number of words to randomly choose for the body of the email.

GenerateEmails.exe then stores the master profile list in an array and begins generating emails and writing them to the input folder in *C:\Users\Victim1\Desktop\MALWARE\input\input.txt*. The number and length of the emails are specified by the first and third arguments of *generateEmails.exe*. Once the email is written to *input.txt*, a copy of the email is written to the text file *email.txt* inside the log folder. The copy of the email is utilized in the validation process. After the email is written to both text files, *generateEmails.exe* executes the sleep command for the number of milliseconds specified by the second argument value of *generateEmails.exe*. After *generateEmails.exe* finishes generating the number of emails specified at execution, the experiment continues until MAPS reaches the next communication interval with the C2 server. At this point the run is concluded and the results are evaluated.

3.4.4.4 Workload

In order to validate MAPS ability to successfully identify and report matches, three different user types are defined with each user type consisting of three factors. Table 3.1 lists the factors for each user.

Table 3.1: Functionality Experiment - Email Generation

User Type	**Total # of Emails to Generate**	**Sleep Timer (ms)**	**Email Length (Words)**
Minimal	100	1000	10
Average	300	500	30
Extreme	500	0	50

The numbers chosen for each user is based upon both domain and design knowledge for this research. While it is important to choose numbers that properly validate MAPS, the ability of MAPS to correctly identify the matches is of primary importance. The extreme user numbers represent well above what MAPS would see in a deployed environment. A profile size of fifty is chosen for all repetitions, and a new random profile is generated for each repetition. This size is chosen based upon previous tests demonstrating that fifty profile words provides sufficient matches to show that MAPS is capable of identifying matches. While it is apparent that a minimal user would not send one hundred emails a second in a production environment, the difference between the Minimal user and the Extreme user are of more interest to this research. The number chosen for the Extreme user of five hundred emails with no delay between email generation is meant to represent a much higher work load then would be expected in a real environment. The sleep timer is set to zero in the Extreme user case, showing that MAPS is capable of ingesting emails as quickly as they can be generated. The Average user variables are chosen to demonstrate numbers halfway between the Minimal user and the Extreme user.

Each user type and its associated variables are considered to be its own test. Each test is then repeated five times. Five repetitions are chosen in order to provide confidence that the accuracies obtained by the experiment are repeatable. At the end of each repetition the logging data associated with that test and repetition combination is saved. The guest operating system for the victim client machine is reset to the beginning experiment system state to prevent any changes that occur during the repetition from affecting subsequent tests. A duplicate database on the C2 server containing match information is created with the name of the test and repetition number (i.e., Minimal_r1 represents the database name for the first repetition of the Minimal user test).

3.4.4.5 Functionality Experiment Evaluation Technique

In order to validate MAPS ability to successfully identify and report matches from a random profile with a random set of emails, several logs are collected on the Victim machine:

- *GenerateEmails.exe* writes each randomly generated email to a text file called *emails.txt* along with the number associated with the order the email is generated.
- Throughout the execution of *maps.exe*, actions that are taken by MAPS are recorded in *log.txt* which include:
  - Each email that is parsed along with the associated number of that email.
  - Each time the call home timer expires, the message that is sent back to the C2 server is recorded.

        - Errors that occur during MAPS execution are recorded.
- The C2 server adds the match words, number of match occurrences, as well as system and user identifying information to the database.

Validation for the functionality experiment is a two-step process:

1. Compare the number of emails reported by MAPS in *log.txt* against the number of emails generated by *generateEmails.exe*. This is to ensure that all the emails generated are parsed by MAPS.
2. Utilize the tool grep to determine the number of profile words that are present in the *log.txt* and *email.txt*. These numbers are then compared with the match numbers existing in the database on the C2 server to validate that the matches are properly found and communicated.

This experiment is designed to show functionality of this proof of concept. The resulting accuracies are analyzed using a 95 percent confidence interval (CI).

3.4.5 Utility Experiment

The utility experiment focuses on the validation of MAPS capability to produce a network attack priority list by utilizing a real email data set and profiles that are meaningful. For the purposes of this research, meaningful profiles are considered to be a list of words that bear a semantic relationship with one another. The profiles that are chosen for this experiment have no pre-existing relationship with the email data set and no correlation is performed to determine which profiles should be used. Other than the emails and profiles used, the experimental setup for the utility experiment is identical to the set up for functionality experiment.

A snapshot of the VM is taken at this point with all applications ready to be executed on the system. This snapshot is entitled Begin. After the chosen profile is uploaded to the C2 server, *startmaps.exe* is executed on the victim machine in order to set up the working environment for *maps.exe*. *Profile.txt* as well as *maps.exe* is downloaded from the C2 server. *Maps.exe* is then executed without any graphic interface or command line on the victim machine, ingesting the downloaded profile.

3.4.5.1 Utility Experiment Profiles

In order to properly validate MAPS' ability to work with meaningful data, outside sources of the profiles are employed. The profiles in this experiment are chosen from the Onelook dictionary website [One12]. Profiles are generated by searching for a master word. Onelook.com consults 100,000+ dictionaries to return a list of words bearing semantic relationships to one another. For the purposes of this research, the list of words returned including the master search word, are considered to be a meaningful profile. Five different profiles are generated for the utility experiment with the following search terms:

1. Business
2. Fraud
3. Love
4. Vacation
5. Family

These topics are chosen for their likelihood to be discussed in a corporation's email system. The top one hundred and fifty terms are taken from the results; terms with multiple words, symbols, and not commonly used words are human edited until fifty words remain with semantically related definitions. For reproducibility, Appendix C includes all five profiles.

3.4.5.2 Email Data Set

The email data set in use for this experiment is a research data set from Enron [Enr03] email servers containing approximately a half a million emails organized by the name of the user. The Federal Energy Regulatory Commission made this data set public during their investigation. The set contains an email folder for one hundred and fifty different users. The utility experiment consists of ten victim machines that are labeled Victim1 through Victim10. Each victim machine contains the emails from one randomly selected user. The Ruby script *parseEmail.rb* is written to parse the content of each email, pulling out data that is useful to this research. This information includes the from address of the user sending the email and the body of the email, which is further edited removing any punctuation, symbols, and additional whitespace. The remaining header and meta information is not processed. The script is run on each user file individually to create a file for each victim as well as run on all email folders collectively to produce the *ex2output* file, containing the parsed emails for all ten users. As a result of this processing, various emails are missing information in either the from field or the body. These emails are excluded from the experiment.

3.4.5.3 Utility Experiment Execution

With *maps.exe* running on the victim system, *writeEmails.rb* writes each email to *input.txt*. The experiment is run with the five different profiles discussed in Section 3.4.4.1. As each repetition completes, the results are stored in a unique database on the C2 server bearing the name of the experiment and repetition number (i.e., ex2r1 for the first repetition of the utility experiment). The victim VMs are reset to the beginning snapshot and the next profile is loaded onto the C2 server.

3.4.5.4 Utility Experiment Evaluation Technique

Validation of the utility experiment consists of determining the percentage of matches that MAPS is able to successfully identify. There are several factors that potentially limit MAPS ability to identify a match; a subset of possible issues are listed below:

- Software discrepancies. Many software bugs have been discovered and remedied during the design process. However, due to time limitations bugs may still exist that affect the results when utilizing the Enron dataset.
- Regex is used to edit the 500,000 emails in the Enron data set. With this size of data set, it is possible that some symbols may not have been adequately sanitized from the emails. This can cause issues with matching.
- Virtual Networking. With virtual networking in use, it is difficult to troubleshoot packet transmission within the internal software of VMware.

Logs that are generated from this experiment exist in the form of database records containing the match information generated throughout the repetition and the logs generated by MAPS as discussed in Section 3.4.3.5. The grep utility is employed to search the email data set for each profile words. Matches that are discovered by the grep utility that are not present in the C2 database or matches in the database that are not discovered by grep are considered to be misses. Conversely, any matches that exist in the C2 database and are revealed by grep are considered to be accurate. Overall accuracy is calculated by:

$$Accuracy = \frac{grep\ hits - MAPS\ misses}{grep\ hits} \tag{3.1}$$

A 95 percent CI is used in order to interpret the results of the utility experiment. Once the results of the experiment have been analyzed, network attack priority lists are created from the results.

3.5 Summary

This chapter discussed in detail the overall design and design considerations for the research tool MAPS to be able to identify matches to profiles and create a network attack priority list. The steps for setting up the development environment in order to develop MAPS and associated helper applications and scripts are discussed in detail. Finally, the testing environment and testing details as well as the validation process of the results are presented.

4. Results and Analysis

4.1 Overview

This chapter provides detailed results and analysis for the experiments outlined in Section 3.4. Section 4.2 and Section 4.3 give detailed analysis of the functionality experiment and utility experiment respectively. Section 4.4 examines characteristics of generating network attack priority lists. A summary of the chapter is presented in Section 4.6.

4.2 Functionality Experiment

The focus of the functionality experiment is to demonstrate MAPS ability to find, record, and report matches to the profile.

4.2.1 Validation Overview

Validation for the functionality experiment is accomplished using the Linux utilities grep version 2.5.1 and wc. Each time *generateEmails.exe* creates an email and writes it to the *input.txt* file, it also creates another copy of that email and stores it in the *\Desktop\log\email.txt* file as well. This *email.txt* file is considered to be the master list of emails for each repetition of the functionality experiment.

4.2.1.1 Validation Scripts

The script *exportMatches.rb* is executed on the C2 server and executes the code in Figure 4.1 to export the database containing the matches found by MAPS into a text file.

```
res = dbh.query "SELECT profileWords.word, SUM(numMatches) as total FROM matches
                 INNER JOIN profileWords ON matches.profileID=profileWords.profileID
                 GROUP BY word;")
res.each do |row|
         outFile.printf "%s\n%d\n", row[0], row[1]
end
```

Figure 4.1: Export Matches MySQL Query

Each match word and its corresponding number of matches found are outputted on consecutive lines for the entire database. A text file, denoted by the name of the database the results are pulled from, is written to the matchFiles folder for each database. The script *ex1Validate.rb* automates the validation process for all repetitions of the functionality experiment simultaneously. This script uses the grep utility to search *email.txt* for each profile word and records the number of times it appears in all randomly generated emails. The profile words and the matchFile for the current repetition are each read into the arrays shown in Figure 4.2.

```
matchFileArray = IO.readlines matchFile
    profileArray = IO.readlines profile
```

Figure 4.2: Functionality Experiment Validation File Arrays

4.2.1.2 Grep String

Grep is then used with each index of the `profileArray` array in conjunction with the wc utility to count each occurrence. Building the command issued to the system is shown in Figure 4.3.

```
command = "grep -E -o '[ |#?]" + profileWord.chomp + " ' " + emailLogPath + " | wc -l"
```

Figure 4.3: Grep Pattern Match String

The –E flag tells grep to interpret the pattern between the single quotes using extended regex, and the –o flag causes grep to only show the part of the line that matches the pattern. The output of grep is then piped into the wc utility with the –l flag which causes the number of lines to be counted. Extended grep is used because basic grep output only outputs the lines containing any matches to the pattern. This causes multiple pattern matches on one line to be counted as one occurrence. The usage of wc as well as the –o flag with extended grep returns an accurate count of multiple matches on a single line. The regex pattern matches either a single space or 0 or 1 occurrences of the `#` symbol followed by the profile word and ending with a single space. The `command` variable is then issued as a system command, shown in Figure 4.4.

```
count = `#{command}`
```

Figure 4.4: System Grep Command

The variable `count` receives the return value when `command` is issued to the system. `count` is a count of the profile words match occurrences.

4.2.1.3 Match Criteria

Each profile word in `profileArray` is then searched for in the `matchFileArray` array. When a match is found `count` is compared to the count in `matchFileArray`. If the number of occurrences match then a hit is recorded for each occurrence. Two cases are considered to be misses.

- The profile word is found by grep but is not found by MAPS client. A miss for each grep count of the profile word is counted.

- Grep and MAPS find a different number of occurrences of a profile word. The difference between the grep count and the MAPS count are recorded as misses (i.e., the profile word ‘business’ is found by grep to have occurred 47 times and MAPS 32 times. This is recorded as 32 hits and 15 misses).

In the event that MAPS either records more occurrences than grep or records a profile word match that grep does not, manual verification is required. Grep is expected to find all occurrences of the profile word, therefore if MAPS finds more matches than grep, it is likely that an error has occurred. Accuracy is determined by Equation 3.1.

4.2.2 Results & Analysis

The functionality experiment consists of five repetitions for each of the three user types resulting in fifteen runs. The 95 percent CI is calculated for each of the three users, and the results are shown in Table 4.1. Due to domain knowledge and knowing the accuracies cannot be over 100 percent, the intervals are capped at 100.

Table 4.1: Functionality Experiment Confidence Intervals

User	Mean Accuracy	95 percent C.I.	Interval
Minimal	100	±0	[100]
Average	99.87	± 0.36	[99.510, 100]
Extreme	99.898	± 0.173	[99.725, 100]

4.2.2.1 Validation Error

In the fifteen runs of the functionality experiment, there are three cases that MAPS produced one more match than grep: Average_r1, Extreme_r4, and Extreme_r5. These cases require manual verification to determine the cause of the additional match found by MAPS. In each case, MAPS correctly identified the matches that are present in the email. Grep does not support positive look-aheads, which allows regex to match a character followed by another character without consuming the second character. Because of this, the case when two of the same match words follow each other can not be detected. For example, if the match word is “tomorrow”, the phrase “tomorrow tomorrow” is not detected. The regex consumes the space between the two words and therefore the second tomorrow does not contain a space followed by the word followed by the space. This results in the pattern not matching the string.

4.2.2.2 Manual Verification

Due to the data set limit of 3075 words, the larger the number of words generated the more likely the case of two of the same word occurring. This explains why it occurs twice in the extreme data set, while not occurring at all in the minimal data set. Despite these discrepancies in validation, the 95 percent CI demonstrates that with random data sets, further runs would result in well over 99 percent accuracy. This experiment successfully demonstrates that even at different user levels MAPS is able to successfully identify the profiles and report the matches back to the C2 server with a verifiable accuracy above 99 percent.

4.3 Utility Experiment

The utility experiment uses an email corpus and meaningful profiles to demonstrate MAPS' ability to accurately identify matches with non-contrived input.

4.3.1 Validation Overview

Validating the utility experiment requires confirming the communication between the MAPS client applications on the ten victim machines and the C2 server. The validation of this data takes place in a similar manner to that of the functionality experiment with the following difference. All ten email files have been consolidated into one single file: *ex2output*. Grep is used on this file just like the functionality experiment with grep matches being considered the correct number of matches. The same grep string, match criteria, and accuracy formula is applied to the results of the utility experiment.

4.3.2 Results & Analysis

The utility experiment consists of 5 different profiles applied to 10 different victim clients. Each victim client contains a unique set of emails to be written. This results in 5 runs of the experiment. Table 4.2 describes the results of the utility experiment.

Table 4.2: Utility Experiment Validation

Run	**Grep Hits**	**MAPS Hits**	**Misses**	**Accuracy**
1	56004	54652	1352	97.586
2	1389	1326	63	95.464
3	9628	9484	144	98.504
4	17508	17236	272	98.445
5	24472	23756	716	97.074

4.3.2.1 Data Verification

All five runs of the utility experiment show accuracies above 95 percent. In order to provide confidence in these accuracies, the first run is repeated three times. The results for all three repetitions of run one are shown in Table 4.3.

Table 4.3: Utility Experiment - Run One Repetitions

Repetition	**Grep Hits**	**MAPS Hits**	**Misses**	**Accuracy**
1	56004	54652	1352	97.586
2	56004	54655	1349	97.591
3	56004	54655	1349	97.591

A 95 percent CI is calculated for the three repetitions. The mean accuracy for the repetitions is 97.589, and the 95 percent CI is 97.586 to 97.591. This shows the mean of additional repetitions for this run should fall between 97.586 percent and 97.591 percent. These repetitions verify that MAPS is able to repeatedly identify a high percentage of matches in the Enron email corpus. Along with the functionality experiment, these results confirm that with a email corpus and meaningful profiles, MAPS is able to effectively match the profiles and report them to the C2 server.

4.4 Network Attack Priority Lists

Network attack priority lists are generated for the systems in the utility experiment. In order to create the network attack priority lists for each run, the matches stored in the C2 database are used.

4.4.1 Building Network Attack Priority Lists

In order to build the network attack priority list, the MySQL query in Figure 4.5 is issued to each run's database:

```
        SELECT system.uname AS User_Name, system.ip AS Internal_IP,
SUM(matches.numMatches) AS Matches FROM system INNER JOIN matches ON
system.ipID=matches.ipID GROUP BY matches.ipID ORDER BY Matches DESC;
```

Figure 4.5: Network Attack Priority List MySQL Query

This query selects relevant information from the database for identifying each system, such as the system username and internal IP address. For each system in the run, the total number of matches are summed together and displayed in the Matches column. The Matches column is sorted in the table from most matches to least matches. Using this query, the network attack priority lists are generated. This list shows the attacker what order to attack systems on the network.

The top two systems for each run of the utility experiment are listed below in Table 4.4 along with the IP address and the number of matches associated with each system.

Table 4.4: Utility Experiment Network Attack Priority Lists - Top 2 Results

Run	Profile Search Word	System	IP Address	Matches
1	Business	Victim5	192.168.87.15	23551
		Victim4	192.168.87.14	11975
2	Fraud	Victim5	192.168.87.15	633
		Victim4	192.168.87.14	309
3	Love	Victim4	192.168.87.14	1879
		Victim5	192.168.87.15	1720
4	Vacation	Victim4	192.168.87.14	4935
		Victim5	192.168.87.15	4161
5	Family	Victim5	192.168.87.15	9487
		Victim4	192.168.87.14	5974

Victim4 and Victim5 have the highest number of matches for all the runs in the utility experiment. This is related to the fact that they have significantly more words in their respective email files than the other victims. In order to obtain matches on uncorrelated data, profiles describing common topics of discussion are chosen. This results in more matches occurring in the emails that contain more words. Since each profile word is considered to be significant, the total number of matches per topic determines the network attack priority list. Refining the profiles could provide a more robust network attack priority list. However, profile generation is beyond the scope of this research.

Table 4.5 shows the network attack priority list for the first run of the utility experiment.

Table 4.5: Utility Experiment – Business Profile

User_Name	**Internal_IP**	**Matches**
Victim 5	192.168.87.15	23551
Victim 4	192.168.87.14	11975
Victim 1	192.168.87.11	4293
Victim 2	192.168.87.12	2848
Victim 9	192.168.87.19	2709
Victim 10	192.168.87.20	2534
Victim 8	192.168.87.18	2459
Victim 3	192.168.87.13	1959
Victim 6	192.168.87.16	1477
Victim 7	192.168.87.17	855

As shown in Table 4.5, Victim5 clearly has the most matches to the business profile out of the ten systems used for run one of the utility experiment. This data provides a basic means of identifying a system on the network that has users that are discussing topics via email relating to business.

4.4.2 Email User Identification

Issuing a further database query results in more detailed information about the users on the system being obtained. Looking at the first run of the utility experiment in Table 4.5, Victim5 has the most hits. By issuing the query in Figure 4.6, a list of email addresses associated with Victim5 and the number of matches for each user is displayed.

```
SELECT emailAddress.fromAddress AS Email, SUM(matches.numMatches) AS Matches FROM matches
INNER JOIN emailAddress ON matches.emailID=emailAddress.emailID INNER JOIN system ON
matches.ipID=system.ipID WHERE matches.ipID=5 GROUP BY matches.emailID ORDER BY Matches;
```

Figure 4.6: Identify Top Email Matches

While this query results in one hundred and thirty-three email address matches, the top four results are likely of the most interest to an attacker looking for matches on the business profile.

Table 4.6: Utility Experiment – Business Profile Breakdown by Top 4 Users

Email Address	Matches
Sarah.palmer@enron.com	15020
m..schmidt@enron.com	3834
Karen.denne@enron.com	1338
Courtney.votaw@enron.com	1313

Table 4.6 shows that 21,505 matches out of the 23,551 matches come from 4 users, with the sarah.palmer@enron.com email address clearly having the majority of the

matches. The attackers can focus their efforts on these users, saving valuable hours and providing a clear starting point to begin harvesting information.

4.5 Summary

This chapter provides details on the results, validation, and analysis of both experiments performed and discussed in Section 3.5. Final results show MAPS ability to identify a high percentage of matches with both a random data set and real world input. With MAPS functionality demonstrated, the utility of MAPS is shown by further querying the match database to determine which systems contribute the most matches for each system. The systems rank from most matches to least matches, giving the attacker a network attack priority list to plan a computer network attack. In addition to the network attack priority list, further analysis of the results show the ability to determine which email users contribute the most to the profile matches. Experimentation successfully demonstrates both the functionality and utility of MAPS.

5. Conclusions and Recommendations

5.1 Overview

This chapter concludes the research presented in this thesis. The following sections discuss the application of this research, future works, and a summary of the thesis.

5.2 Significance of Research

People have grown accustomed to communicating electronically for many different reasons. With limited resources and technical proficiencies available, it is important to utilize skills in the most efficient manner as possible. This research highlights a manner in which automation can be employed to focus limited resources in other areas. The use of profiles to describe information can be created by non-technical analysts, while attackers can more efficiently attack the network. This research shows that information profiles aid in the targeting of computers on a network without human interaction on the systems.

This research successfully demonstrates that a ranking of computer targets by comparing real time user email traffic from the perspective of the users workstation against pre-determined profiles can be implemented.

5.3 Recommendations for Future Research

This thesis is a small part in the research to increase the efficiency of attackers on computer networks. Some proposed modifications to the implementation are as follows:

- Increasing the complexity of the profiles utilized to determine activity on the system. While this research did not focus on how the profiles are generated, the utilization of more specific profiles will eliminate many false positives and negatives. For this research to be employed in an actual attack operation, profiles would need to be more specific to the type of information that is being sought.
- A much higher presence will be necessary on the target system. In order to truly have situational awareness of what is occurring on the system, it is necessary to have more information then what can be obtained via email alone. Activities such as web browsing, document creation, remote access logs, etc. can indicate the information residing on the workstation.
- A more robust means of parsing the content. This research focuses on text-based content that could be compared to a text based profile. With multiple means of gathering data, research on content parsing to produce meaningful results is necessary. In addition, the parser must properly handle errors.
- Increased stealth components. While MAPS employs some basic customization options for the network being attacked, stealth components would be required for a deployable tool. Rootkits, covert transmission mediums, and a decreased executable size reduces the likelihood of discovery.

5.4 Summary

This research identifies means with which an attacker can increase the efficiency of operations against computer information systems, without having to increase their technical work force. This is achieved by defining and implementing the following steps:

1. Collect email traffic on a windows workstation and compare the emails to a predefined profile sending the matches to the C2 server. MAPS software is implemented using the C programming language and runs on the Windows operating system.
2. Verify the ability to successfully identify and report matches from a random profile with a random set of emails. Grepping the contents of the email output files and comparing these results against the C2 database determines MAPS ability to identify matches.
3. Verify the research goal by loading semantically-related profiles and utilizing the Enron email corpus [Enr03] to determine if a network attack priority listing can be derived. Profiles generated by searching onelook.com for topics are run against the email data set from the Enron corporation. Validation takes place by demonstrating that MAPS is able to correctly identify 90 percent of the matches and MySQL queries are used to present the information via a network attack priority list.

This research succeeded in realizing the goals and demonstrates a foundation for automating the collection of information from a system for the purpose of determining the topics of discussion of the users on the system.

Appendix A: Functionality Experiment Results

Run	Grep hits	MAPS Hits	Misses	Accuracy
Minimal_r1	13	13	0	100
Minimal_r2	11	11	0	100
Minimal_r3	12	12	0	100
Minimal_r4	14	14	0	100
Minimal_r5	10	10	0	100
Average_r1	154	155	1	99.351
Average_r2	152	152	0	100
Average_r3	137	137	0	100
Average_r4	161	161	0	100
Average_r5	146	146	0	100
Extreme_r1	365	365	0	100
Extreme_r2	385	385	0	100
Extreme_r3	374	374	0	100
Extreme_r4	394	395	1	99.746
Extreme_r5	392	393	1	99.745

Appendix B: Experiment Confidence Intervals

Functionality Experiment: n=5

Run	Mean	SD	SE	CI	Range
Minimal	100	0	0	0	100
Average	99.8702	0.29024162	0.1298	0.36038257	99.5 – 100.2
Extreme	99.8982	0.13939584	0.06233971	0.1730828	99.7 – 100.1

Utility Experiment Run 1: n=3

Mean	SD	SE	CI	Range
.97589	3.0927E-05	1.7856E-05	7.6828E-05	0.97581-0.97597

Appendix C: Utility Experiment Profiles

Run 1	Run 2	Run 3	Run 4	Run 5
Business	**Fraud**	**Love**	**Vacation**	**Family**
account	bite	adoration	accomplished	ancestry
activity	charlatan	adore	appeal	baby
advertising	cheat	affection	away	blood
affair	collateral	agape	ball	blue
affairs	colluder	amorous	beach	brood
agency	conveyance	ardor	condos	category
bank	cross	attachment	cottage	children
banking	deceit	attraction	country	clan
brokerage	deception	bang	cruise	class
business	defraud	bed	dacha	common
byplay	defraudment	beloved	down	descended
carrier	dishonesty	bonk	ego	descent
clientele	dupery	charity	eholiday	domestic
commerce	election	cherish	fab	dynasty
commercial	extrinsic	crush	fabulous	familial
company	fake	dear	furlough	families
competition	faker	dearest	getaway	family
concern	fiddle	devotion	honeymoon	feline
construction	fleece	dote	idyllic	fellowship
corp	foist	enjoy	kiss	folk
corporation	fraud	feeling	leave	genealogy
custom	fraudless	fornicate	leisure	genera
deal	fraudulence	f***	look	group
enterprise	fraudulent	heart	midsummer	heirloom
establishment	gyp	honey	mind	home
firm	hoax	hump	month	homefold
house	hustle	infatuation	nag	hominid
industrial	imposter	intrigue	outing	house
industry	impostor	jazz	pass	household
maker	ingannation	know	picnic	ilk
management	intrinsic	liking	plate	kin
manager	mulct	love	playground	kinfolk
manufacturer	namedropper	loved	rainy	kinsfolk
manufacturing	pretender	loveless	recess	kinsperson
mercantile	pseudo	lovely	rental	line
occupation	pull	lovemaking	resort	lineage
office	racket	lover	resorts	marriage
open	ringer	loves	rested	matriarch

operation	scam	loving	sightsee	menage
organization	sham	passion	sixthly	name
partnership	shark	passionate	spa	parentage
patronage	surreptitious	patriotism	spend	parents
practice	swindle	philanthropy	subject	patriarch
publishing	trickery	platonic	then	people
sector	trickster	romance	throw	related
shop	trumpery	romantic	trips	sept
stage	unmasking	sc***	vacation	shrew
trade	victimize	strong	vacationed	stock
tycoon	watcher	unrequited	vacationer	taxonomic
work	wrench	worship	vacationing	tribe

Bibliography

[Bar10] S. Barish. “Passive Network Analysis”. 2010. http://www.symantec.com/connect/articles/passive-network-analysis. Retrieved 11 August 2012.

[Coy11] S. Coyne. “The Getaway: Methods and Defenses for Data Exfiltration”.. https://media.blackhat.com/bh-dc-11/Coyne/BlackHat_DC_2011_Coyne_Gateway-wp.pdf. Retrieved 11 August 2012.

[Dam12] DameWare. http://www.dameware.com/. Retrieved 11 August 2012.

[Dsn12] Dsniff. http://monkey.org/~dugsong/dsniff/. Retrieved 11 August 2012.

[Enr03] Enron. http://www.cs.cmu.edu/~enron/. Retrieved 11 August 2012.

[Ett12] Ettercap. http://ettercap.sourceforge.net/. Retrieved 11 August 2012.

[Far12] Farmanager. http://www.farmanager.com/. Retrieved 11 August 2012.

[GBC06] A. Giani, V. Berk, G. Cybenko. “Data Exfiltration and Covert Channels”. http://www.ists.dartmouth.edu/library/293.pdf. Retrieved 11 August 2012.

[Gho10] A. Ghosh. “The Reign of Zeus”. https://www.invincea.com/blog/tag/exfiltrating-data/. Retrieved 11 August 2012.

[Gje10] T.Gjelten “Cyberwarrior Shortage Threatens U.S. Security”. http://www.npr.org/templates/story/story.php?storyId=128574055. Retrieved 11 August 2012.

[Hel11] Help Net Security. “Malware-driven pervasive memory scraping”. http://www.net-security.org/malware_news.php?id=1641&utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+HelpNetSecurity+%28Help+Net+Security%29. Retrieved 11 August 2012.

[HoB06] G. Hoglund, J. Butler. 2006. *Subverting the Windows Kernel Rootkits*. Upper Saddle River, NJ: Addison-Wesley.

[Hus11] B. Hussey. "Decoding Data Exfiltration – Reversing XOR Encryption". http://crucialsecurityblog.harris.com/category/malware-reverse-engineering/. Retrieved 11 August 2012.

[IAL09] J. Iglesias, P. Angelov, A. Ledezma, A. Sanchis. 2012. "Creating Evolving User Behavior Profiles Automatically". *IEEE. IEEE* Transactions on Knowledge and Data Engineering. Volume 24 Issue 5. pg 854-867.

[Ide10] Identity Finder. "Data Loss Prevention: Data-at-Rest vs. Data-in-Motion". http://www.identityfinder.com/us/Files/WhitePaper.pdf. Retrieved 11 August 2012.

[LAH11] M. Ligh, S. Adair, B. Hartstein, M. Richard. 2011. *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*. Indianapolis, Indiana: Wiley Publishing Inc.

[Lan07] B. Lane, "The USAF Standard Desktop Configuration (SDC)". http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCwQFjAA&url=http%3A%2F%2Fdownload.microsoft.com%2Fdownload%2F5%2Fc%2F4%2F5c46c4a0-950f-40a9-9a8f-9af4a2869bc2%2FWhitePaper_FDCC AirForce.doc&ei=FPrMTuK9CefV0QGP3fUy&usg=AFQjCNFY0N382. Retrieved 11 November 2011.

[Lop06] K. Lopez. "Standard Desktop Configuration keeps AFMC ahead of 'bad guys'". http://www.afmc.af.mil/news/story.asp?id=123020383. Retrieved 11 August 2012.

[Man12] Mandiant. 2011. "Mandiant ApateDNS". http://www.mandiant.com/resources/download/research-tool-mandiant-apatedns. Retrieved 11 August 2012.

[Mes11] E. Messmer. "Memory scraping malware goes after encrypted private information". http://www.networkworld.com/news/2011/022211-pervasive-memory-malware.html. Retrieved 11 August 2012.

[Mil11] S. Miller. "Pervasive Memory Scraping a Growing Threat that Bypasses Encryption". http://www.processor.com/articles//P3313/14p13/14p13.pdf?guid. Retrieved 11 August 2012.

[Msd112] MSDN. http://msdn.microsoft.com/en-us/library/ms717422.aspx. Retrieved 11 August 2012.

[Msd212] MSDN. http://msdn.microsoft.com/en-us/library/windows/desktop/aa364417(v=vs.85).aspx. Retrieved 11 August 2012.

[MSK09] S. McClure, J. Scambray, G. Kurtz. 2009. *Hacking Exposed 6*. New York: McGraw-Hill Companies.

[Net12] Netmarketshare. http://www.netmarketshare.com/os-market-share.aspx?qprid=9. Retrieved 11 August 2012.

[Ntu11] DameWare NT Utilities. “DameWare NT Utilities”. http://dameware.info/products/dntu/. Retrieved 11 August 2012.

[OBB06] J. Onroy, J. Becerra, F. Bellas, R. Duro, F. López-Peña. 2006. “Automatic Profiling and Behavior Prediction of Computer System Users”. *IEEE*. *IEEE* International Workshop on Measurement Systems for Homeland Security, Contraband Detection and Personal Safety. pgs. 62-66.

[One12] Onelook. http://www.onelook.com/. Retrieved 11 August 2012.

[Ope12] Openwall. http://www.openwall.com/john/. Retrieved 11 August 2012.

[PeI09] N. Percoco, J. Ilyas. “Malware Freakshow”. Defcon 17. http://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-nicholas_percoco-jibran_ilyas-malware_freak_show.pdf. Retrieved 11 August 2012.

[Per10] N. Percoco. “Data Exfiltration: How Data Gets out”. http://www.computerworld.com/s/article/9169978/Data_Exfiltration_How_Data_Gets_Out?taxonomyId=83&pageNumber=2. Retrieved 11 August 2012.

[PIM11] N. Percoco, J. Ilyas, R. Merritt. “Global Security Report 2011”. http://immersionltd.com/Immersion/documents/Trustwave_WP_Global_Security_Report_2011.pdf. Retrieved 11 August 2012.

[Pro11] Programming4us. “Windows 7: Indexing Your Computer for Faster Searches (part 2) - Specifying Files Types to Include or Exclude”. http://programming4.us/desktop/2346.aspx. Retrieved 11 August 2012.

[PSI10] N. Percoco, C. Sheppard, J. Ilyas. "Evolution of Malware: Targeting Credit Card Data in Memory". http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&ved=0CD0QFjAE&url=http%3A%2F%2Fwww.trustwave.com%2Fdownloads%2Fwhitepapers%2FTrustwave_WP_Evolution_of_Malware_.pdf&ei=U2jiTsjWFsXe0QGhhJT1BQ&usg=AFQjCNHCnEXUfhPbKE9aTnC20Zp8LWKfKQ&sig2=AbCBFjB7ddpaFe89WfokMQ. Retrieved 11 August 2012.

[RBC07] D. Robinson, V. Berk, G. Cybenko. "Online Behavioral Analysis and Modeling Methodology (OBAMM)". http://www.ists.dartmouth.edu/library/414.pdf. Retrieved on 11 August 2012.

[Rob10] D. Robinson. 2010. "Cyber-Based Behavioral Modeling". Dissertation. Dartmouth College. http://dms.sagepub.com/content/9/3/195.abstract. Retrieved 11 August 2012.

[Sea00] Searchwindowsserver. "domain controller". http://searchwindowsserver.techtarget.com/definition/domain-controller. Retrieved 11 August 2012.

[Sha07] D. Shackleford. "Regulations and Standards: Where Encryption Applies". http://www.sans.org/reading_room/analysts_program/encryption_Nov07.pdf. Retrieved 11 August 2012.

[SkL06] E. Skoudis, T. Liston. 2006. *Counter Hack Reloaded 2nd Edition*. Upper Saddle River, NJ: Prentice Hall.

[TuS03] I. Tutănescu, E. Sofron. "Anatomy and Types of Attacks against Computer Networks". 2nd Roedunet International Conference 2003. http://193.226.6.174/roedunet2003/site/conference/papers/TUTANESCU_I-Anatomy_and_Types_of_Attacks_against_Computer_..pdf. Retrieved 11 August 2012.

[Vis08] Visa. "Visa Data Security Alert". http://usa.visa.com/download/merchants/debugging_software_memory.pdf. Retrieved 11 August 2012.

[VSP10] S. Venter, C. Sheppard, N. Percoco. "POS Memory Parsing Malware Briefing Attacks on Kernel Memory". http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCEQFjAA&url=https%3A%2F%2Fwww.trustwave.com%2Fdownloads%2Fspiderlabs%2FTrustwave_SpiderLabs_Briefing_POS_Malware_Attacks_on_Kernel_Memory_June_2010.pdf&ei=mBb6TvHXF-

Hf0QHylsHgAQ&usg=AFQjCNHy71IdbuVY7686wm3jBEsNLAlBAg&sig2=IW7tcXh0_D8h9Bf5rzPylA. Retrieved 11 August 2012.

[Wes10] K. Westphal. “Steganography Revealed”. http://www.symantec.com/connect/articles/steganography-revealed. Retrieved 11 August 2012.

[Win11] Windows. “Windows Search”. http://windows.microsoft.com/en-US/windows7/products/features/windows-search. Retrieved 11 August 2012.

[Win12] WinSCP. http://www.winscp.net/eng/download.php/. Retrieved 11 August 2012.

[Wir12] Wireshark. http://www.wireshark.org/. Retrieved 11 August 2012.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 13-09-2012	**2. REPORT TYPE** Master’s Thesis	**3. DATES COVERED** *(From – To)* March 2011 – September 2012

4. TITLE AND SUBTITLE Creating Network Attack Priority Lists by Analyzing Email Traffic Using Predefined Profiles	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Merritt, Eric J., Civ, 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/ENG) 2950 Hobson Way, Building 640 WPAFB OH 45433-8865	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCO/ENG/12-19
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
DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

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
Networks can be vast and complicated entities consisting of both servers and workstations that contain information sought by attackers. Searching for specific data in a large network can be a time consuming process. Vast amounts of data either passes through or is stored by various servers on the network. However, intermediate work products are often kept solely on workstations. Potential high value targets can be passively identified by comparing user email traffic against predefined profiles. This method provides a potentially smaller footprint on target systems, less human interaction, and increased efficiency of attackers. Collecting user email traffic and comparing each word in an email to a predefined profile, or a list of key words of interest to the attacker, can provide a prioritized list of systems containing the most relevant information.
This research uses two experiments. The functionality experiment uses randomly generated emails and profiles, demonstrating MAPS (Merritt’s Adaptive Profiling System) ability to accurately identify matches. The utility experiment uses an email corpus and meaningful profiles, further demonstrating MAPS ability to accurately identify matches with non-random input. A meaningful profile is a list of words bearing a semantic relationship to a topic of interest to the attacker.
Results for the functionality experiment show MAPS can parse randomly generated emails and identify matches with an accuracy of 99 percent or above. The utility experiment using an email corpus with meaningful profiles, show slightly lower accuracies of 95 percent or above. Based upon the match results, network attack priority lists are generated. A network attack priority list is an ordered list of systems, where the potentially highest value systems exhibit the greatest fit to the profile. An attacker then uses the list when searching for target information on the network to prioritize the systems most likely to contain useful data.

15. SUBJECT TERMS
network attack, security, network reconnaissance, profile

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Barry Mullins, Civ, USAF ADVISOR
REPORT U	**ABSTRACT** U	**c. THIS PAGE** U	UU	98	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255 - 3636, ext 7979 barry.mullins@afit.edu