Large Scale DNS Traffic Analysis of Malicious Internet Activity with a Focus on Evaluating the Response Time of Blocking Phishing Sites

by

Jonathan M. Spring

BPhil, University of Pittsburgh, 2008

MSIS, University of Pittsburgh, 2010

Submitted to the Graduate Faculty of

School of Information Sciences in partial fulfillment

of the requirements for the degree of

Master of Science in Information Science

University of Pittsburgh

2010

UNIVERSITY OF PITTSBURGH

School of Information Sciences

This thesis was presented

by

Jonathan M. Spring

It was defended on

April 21st, 2010

and approved by

Sidney Faber, Analyst, CERT Coordination Center

Dr. James Joshi, PhD, Associate Professor

Dr. David Tipper, PhD, Telecommunictaions Program Chair & Associate Professor

Co-Chair: Dr. Prashant Krishnamurthy, PhD, Associate Professor

Co-Chair: Edward Stoner, Analyst, CERT Coordination Center

Large Scale DNS Traffic Analysis of Malicious Internet Activity with a Focus on Evaluating the Response Time of Blocking Phishing Sites

Jonathan M. Spring, M.S.

University of Pittsburgh, 2010

This thesis explores four research areas that are examined using DNS traffic analysis. The tools used for this analysis are presented first. The four topics examined are domain mapping, response time of anti-phishing block lists to find the phishing sites, automated identification of malicious fast-flux hosting domains, and identification of distributed denial of service attacks. The first three approaches yielded successful results, and the fourth yields primarily negative lessons for using DNS traffic analysis in such a scenario. Much of the analysis concerns the anti-phishing response time, which has yielded tentative results. It is found that there is significant overlap between the automatically identified fast-flux sites and those sites on the block list. It appears that domains were being put onto the list approximately 11 hours after becoming active, in the median case, which is very nearly the median lifetime of a phishing site. More recently collected data indicates that this result is extremely difficult to verify. While further work is necessary to verify these claims, the initial indication is that finding and listing phishing sites is the bottleneck in propagating data to protect consumers from malicious phishing sites.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

There are many people who were necessary to make this thesis possible. I would like to thank Sidney Faber for giving me a chance work for him and with this data as well as for his guidance. Huge thanks are due to Ron Bandes and Ed Stoner for their work in designing and writing tools that were instrumental in this analysis. Thanks also to Rhiannon Weaver, Markus DeShon, and the rest of the CERT NetSA team for their help and expertise. Thanks are of course due to Dr. Prashant Krishnamurthy for his time and advice, as well as Dr. James Joshi and Dr. David Tipper for graciously serving on the review panel and providing their insights. Thanks also to my father for his comments, contributions, and support.

1.0 INTRODUCTION

The primary purpose of this thesis is to demonstrate and describe novel analysis that is possible using passive Domain Name System (DNS) traffic analysis. Prior to this research, passive DNS gathering was primarily used for domain mapping. In this technique researchers attempt to correlate, or map, domain names to Internet Protocol (IP) addresses and how these change over time. This thesis demonstrates further applications of passive DNS traffic analysis such as inspecting the speed of identification of phishing attacks. Passive gathering is desirable in security research because the researcher need only listen to the traffic going across the network and does not need to initiate queries or otherwise leave any trace of her activity for those being investigated. DNS is a desirable protocol for analysis because it is both ubiquitous and rich with information, while remaining concise.

The remainder of chapter 1 presents pertinent background information. Section 1.1 will discuss the elements of DNS relevant to this thesis and point to additional sources. The next section will introduce the various goals of the research, and section 1.3 will describe the data with which the analysis worked.

Chapter 2 describes noteworthy software tools that were used to perform the analysis. 2.1 describes Ncaptool, a tool developed by the Internet Systems Consortium (ISC). 2.2 describes updates to Ncaptool that were made in order to facilitate the analysis.

Section 2.3 describes the non-obvious shell scripts that are used to collect, manage, and maintain the DNS data.

The results of the various analysis goals are presented in part 3. First, novel results achieved with mapping different top-level domains are presented. Section 3.2 presents the primary results of the study, which analyze the effectiveness of anti-phishing block lists on the lifetime of a phishing website. In a topic that is closely related to anti-phishing due to one of the manners in which malicious sites are hosted, section 3.3 describes an algorithm to detect fast-flux websites automatically. The final analysis section, 3.4, describes attempts to detect distributed denial of service (DDoS) attacks using DNS information.

Part 4 indicates directions of potential future work in this area. Part 5 offers a summary of the findings and some conclusions to be drawn from them. Finally, for a glossary of terms and acronyms, the reader may consult Appendix A. For a compendium of most of the scripts used in the course of the research, consult Appendix B.

1.1 ABOUT DNS

The Internet is supported by an addressing scheme and packet delivery protocol known as the Internet Protocol. Each host has a unique thirty-two-bit address. For example, the machine I am sitting at has the address 10001000-10001110-01111001-01001101, with dashes added for readability. Early on in the development of the Internet, many of the developers were able to remember these numbers by converting the binary byte values to decimal. This is called the dotted decimal notation and the address above has the dotted decimal value 136.142.121.77. Through the 1970’s, it was possible for human users to organize and remember the numbers for several machines and to memorize “domains” – e.g. all machines at the University of Pittsburgh begin with 136.142. However, as the number of machines grew into the millions and more casual users became involved, something more was needed. What was developed was a mapping between human consumable names and IP addresses. The Domain Name System (DNS) is the standard that provides for this mapping. It allows machine names such as “augment.sis.pitt.edu” to be mapped to the IP address and the location of a host. DNS also provides a means by which a name can be easily reassigned to a new address should that need arise. The DNS protocol dates back to 1983 and RFCs 882 and 883 (Mockapetris 1983). (DNS and Bind, by Cricket Liu and Paul Albitz, provides a detailed description of the protocol and its evolution.)

DNS is administered through a hierarchy of servers that store Domain name data, keep it up-to-date, and respond to requests. The hierarchy divides responsibility for DNS into smaller and smaller chunks. The root servers for “edu” keep track of the servers responsible at the next level down e.g. pitt. The pitt.edu name servers keep track of the name servers at Pitt e.g. the name server for sis – “sis.pitt.edu”. At the highest level of the hierarchy are the root name servers, of which there are 13. These 13 servers answer queries about the top-level domains such as .com, .edu, and .uk; no more general DNS data base exists, and so these servers are the “root” or base of the DNS hierarchy. Although there are 13 logical root name servers, multiple physical servers exist in a distributed architecture, e.g., there are upwards of 200 machines answering queries at the root level (www.root-servers.org). The global DNS architecture is designed to shield higher-level servers from traffic, since personal computers and every DNS server caches and stores the answers it gets for a period of time to keep it from having to send a DNS request every time it needs to know where a certain web address is. Even so, the root servers receive a significant amount of traffic. The H root server experienced an average of 8500 queries per second during a test carried out as part of the "A Day in the Life of the Internet" (DITL) project (Kash 2008).

The information contained in the millions of DNS requests can be quite valuable. DNS packets are of variable size, and are generally limited technically by the size of the User Datagram Protocol (UDP) packet in which they are sent. In practice, they are observed to average about 200 bytes per packet. They contain several IP addresses, domain name text, timestamps, several flags, and some information about where this information came from. There are three entities about which one can gather information by looking at DNS packets: the initiator, the target, and the subject of the query. A DNS request contains the IP addresses of the requestor and the DNS server as well as the target's domain name. The DNS response will contain the request and the answer, which is often an IP address, although other types of information can be shared via DNS as well. While the goal of DNS is to provide information about the subject, the other information in the packets can be analyzed in aggregate to provide network intelligence information. The packet includes information about the name server or name servers that are responsible for being the authority on the subject's information. This information sheds light on the initiator and target as well, such as who knows what about whom, when and how often questions are asked, and who answers for those who hold the authoritative knowledge about a site. This kind of analysis is often referred to as a traffic analysis and is the basis for our research into evaluating the speed of identification of phishing attacks.

1.2 RELATED PRIOR WORK AND INTENDED CONTRIBUTIONS

Various research has been conducted on the state of phishing and malicious activity on the internet. There is solid research into how data breaches and hacking tend to occur (Baker 2008). In the same year, Cyveillance published a white paper detail the costs and damages caused by phishing attacks (Cyveillance 2008). Furthermore, there is evidence that takedown and blocking efforts have a significant effect on the lifetime of phishing sites and on phishing activity (Moore 2007). There exist reliable numbers about the total lifetime of phishing sites, from when they go live until when they are taken down (Aaron 2009). The Anti-phishing Working Group has published four such reports since 2007. It is established that phishing is damaging and that takedown has a significant effect on reducing this damage, however precise numbers as to how long it takes the takedown process to get started are unknown. The primary focus of this thesis is to evaluate the ability to derive such numbers using passive DNS traffic analysis and contrast them with those for the lifetimes of phishing sites on average.

The thesis also touches on other novel results achieved through use of DNS traffic analysis that could also be derived from the same data source as the phishing response time analysis. This data source itself presents a non-trivial database and data analysis difficulty due to the volume of data available; see section 1.4 for details. This thesis does not present the first DNS traffic analysis of malicious network activity. Passive analysis of DNS traffic collected at high traffic routers has been used to detect malicious flux networks outside of email spam lists (Perdisci 2009). This is closely related to work presented in section 3.3. DNS traffic has also been used to build relational databases to generate IP to domain name mappings of various domains (Weimer 2005). The uses of such databases relate closely to section 3.1, however the approach taken to store the data in our approach was much more simple, arguably more robust, and does not seem to have disadvantages to the relational database approach in terms of performance speed, which neither hold as a focus of the work.

It was initially believed a database would need to be engineered to efficiently store multiple weeks of DNS information. This was demonstrated to be untrue early on when it was recognized that the native DNS format was by far the most compact and data rich format. In order to compact the data at all, many flags and bits of information would have to be discarded that had unknown potential value to this or future research. Furthermore, whereas data storage of flat, albeit compressed, DNS packets within Ncap files could be set up with little expertise in the matter of a few weeks, engineering of a complex database to deconstruct and extract data from the DNS packets into a relational database is expected to take the better part of two years, if a current project stays on track. The work in this thesis therefore employs the native DNS format.

If worldwide phishing is to be monitored, the entirety of worldwide web must be monitored. The only tenable protocol with which to do this is DNS, because other ubiquitous protocols, such as TCP/IP, either don’t provide sufficient information about the actors involved or are far to voluminous to allow for anywhere near real time analysis of the ever-changing landscape, such as HTTP. Given these limitations, it is important to learn how to analyze DNS data because it is the best target for this very valuable real time analysis of the Internet. DNS does have various shortcomings, which vary depending on the analysis, that are presented throughout this thesis. This thesis aims to demonstrate the benefits of DNS traffic analysis for answering various questions about malicious activity on the Internet, particularly phishing, and to advance the state of this analysis by introducing a novel method of correcting for the shortcomings in DNS analysis of phishing sites. It also will demonstrate some of the benefits of retaining the native DNS packet format in terms of increases in analysis detail over previous work that used relational databases to store DNS data.

1.3 ANALYSIS GOALS

Initial goals of the project were simply to establish the technical infrastructure to allow analysis to be done. The Security Information Exchange (SIE) provides a community driven data source consisting entirely of observed DNS packets. Section 1.4 describes the data stream, which needed to be captured, stored, and rendered searchable. In order to facilitate useful analysis, this would have to be done in near real time, because of the dynamic and volatile nature of addressing in the global computer network. This was the primary initial goal of the project, but due to the the amount of data able to be stored using the native DNS format and simple linux utilities it was accomplished rather quickly.

The primary goal of the research then became to determine how effective anti-phishing block lists were at getting malicious sites onto their lists. DNS would provide a manner of checking this because one could, hypothetically, discover the earliest detected instance of a request for the malicious site and compare that timestamp to the time at which the site is added to the block list.

Another goal, more established in DNS research generally, was to attempt to describe the structure of a domain passively. Here, the level of detail that could be achieved and the simplicity of the acquisition were in question.   Also an easily accessible and viewable interface was needed.

A goal that was added to the project partway through was to use this DNS data to identify fast flux hosting in the data. Fast flux hosting is a technique of hosting a service at a rapidly changing series of IP addresses. The domain name in question is generally constant while the IP address is often changed at least once every five minutes (www.spamhaus.org). This is of particular interest because this practice is highly correlated with malicious activity by the site being hosted, as it is an effective way to avoid detection and blocking.

Since very little work had been done attempting to utilize DNS as a data source besides the passive domain mapping, anything else novel was to be pursued in order to discover what information it might yield (Faber 2009). This led to attempts to unearth information about DDoS attacks, such as the one targeting South Korea and United States government sites in early July 2009, while data was being collected from the SIE DNS data stream. This was also a motivation behind the attempt at a novel passive domain map.

1.4 ABOUT SIE DATA

As far as the collection infrastructure of the SIE data feed it is worthwhile to directly quote the official description:

> “SIE maintains collection infrastructure for several sets of recursive nameservers around the Internet. The nameservers query authoritative nameservers on behalf of their customers' queries and the answers returned from the authoritative nameservers to the recursive nameservers are captured off the network interface and batch-

> forwarded to SIE. At an SIE rebroadcast facility, the packets from uploaded batches are replayed." (ISC 2009)

Although the location and identity of these collection points is considered sensitive information, each collection point is given a random unique identifier. This allows us to count the number of collection points, which number about a dozen. This low number raises some questions about how representative of a sample the SIE data is of the Internet as a whole. However, without further information about where these collection points are and how interconnected they are it is impossible to know exactly what biases are in one's data. Section 3.1 presents some evidence that the sample is representative.

Some filters are applied to the data as it wends its way towards information subscribers' sensors on SIE's rebroadcast facility. At the sensor point a Berkley Packet Filter (BPF) is applied that only allows valid DNS responses to be captured. The filter applied is:

```
"(dst host NSNAME and udp src port 53 and
udp[10] & 0xf8 = 0x80 and udp[11] & 0xf = 0)
or
(src host NSNAME and icmp[icmptype] = icmp-unreach)" (ISC 2009)
```

This filter ultimately has broad repercussions on what work we are able to do. Through this filter the SIE's primary channel receives approximately 10,000 packets per second. This is channel 2, which is further filtered and reprocessed and rebroadcast onto 12 other channels, each with its own different filter and purpose. A channel is basically a virtual private network established for the purpose of segregating the various sets of data created by applying the various filters. Each channel is a live feed of the DNS packets passing through the Internet, delayed perhaps a few hours depending on the type of filtering that is applied to it.

This research is primarily concerned with channel 5, which is a version of channel 2 that has been deduplicated by suppressing duplicate Resource Record (RR) sets over a 4-hour time window and incrementing a counter to indicate how often the RR set was suppressed. According to the provider, this reduces the traffic flow from 10,000 packets per second to about 2,000, and is the channel recommended by SIE for monitoring passive DNS (ISC 2009). During data collection channel 5 was observed to exhibit a data flow of about 3,100 packets per second, yielding just under 5 Mb per second. This was observed to produce roughly 56 GB of data per day, which could be reduced to 17 GB per day simply by applying the gzip utility.

Several of these design choices are excellent for passively mapping out huge swaths of IP-domain name pairs. However, they are not ideal for every DNS-based research undertaking and have hampered the research presented in this thesis in certain ways. For example, the decision to include only valid DNS packets that are responses, and not queries nor "no such domain" error responses has several repercussions. No data is lost about the query in the response packet because a response includes all the information in the query so that the initiator can identify what question is being answered. However for certain applications the information that could be gleaned from unanswered or malformed queries would be very valuable, and by some estimates only 2% of queries to the root servers are well-formed and answerable (Wessels 2003). The SIE filter is therefore a double-edged sword. It would be splendid to be able to have certain ill-formed DNS queries accessible to the research, however it is probably the case that the data rate would then become unmanageable. Furthermore, the SIE data is the only source of its kind as it is, and so the research must cope with the data available and be aware of the limits of the questions one may ask of the data.

2.0 SOFTWARE TOOLS

In any research one can only delve as far as her tools allow. The previous section described the information architecture with which our tools can work. And while the focus of this thesis is not to describe advances in software engineering, it is important to have a sense of what tools are available and what their limitations are in understanding this research, planning future research, and planning any improvements to the tools of the trade necessary for that research.

This chapter will focus primarily on Ncaptool. This program is used to collect the DNS data at the collection points, rebroadcast within the SIE, apply filters both natively and through plugins, and format the data for human readability. Section 2 relates the improvements that were made to `ncaptool` during the course of this research, primarily by Ron Bandes. Section 3 contains the bash scripts that are used to manipulate both the data and `ncaptool` to make the analysis possible. These scripts are available in Appendix B.

2.1 NCAPTOOL

`Ncaptool` is the primary tool used in this research because it is the primary tool used by the SIE is collecting DNS data. It is available at ftp://ftp.isc.org/isc/ncap/. Given the amount of data that it has been designed to handle, its design is focused on efficiency. It is architecturally reliant upon `libncap`, a stable build of core functions. Ncaptool interfaces directly with this library in order to provide further functionality to the user by allowing a wider range of options including the ability to include plug-in modules. Figure 1 is a representation of all the input and output types for the architecture of the tool.

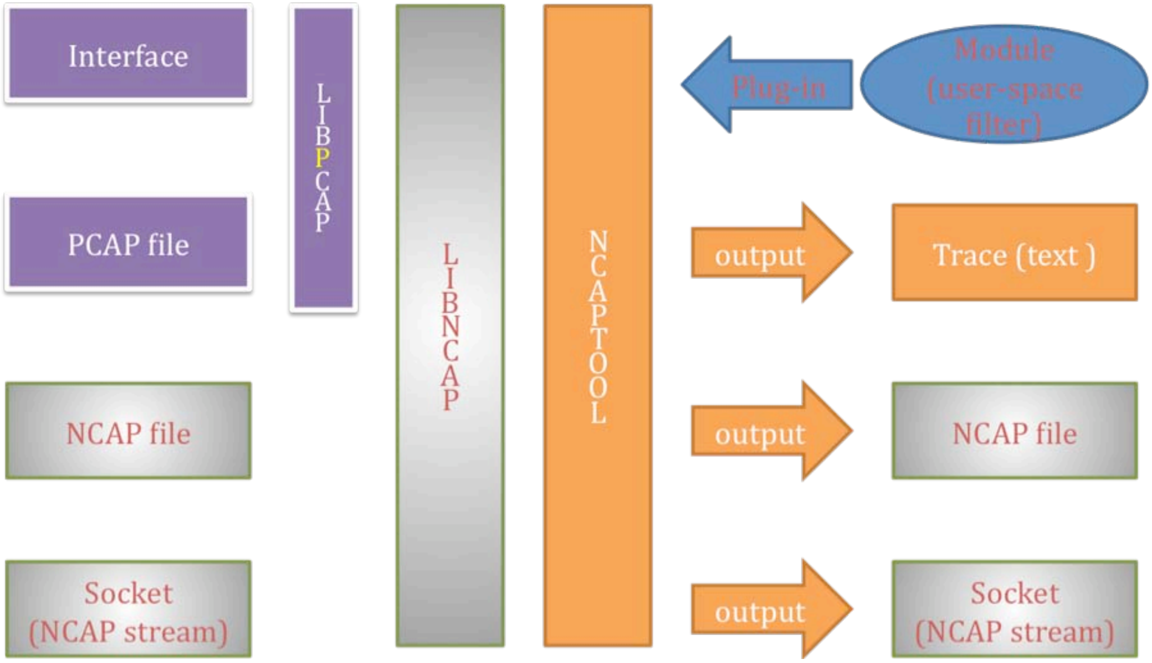


Figure 1: NCAP sources and sinks

Ncap also focused on ameliorating some of the issues with its predecessors, Pcap and DNScap. For example, Ncap incorporated nanosecond resolutions on its timestamps, improving upon Pcap's microsecond resolution because it had been observed to be insufficiently precise. Ncap provided for filtering besides what could be provided by incorporating a BPF filter, even though it still utilizes BPF when possible. These DNS filters allow various DNS message fields to be specified, such as flags, initiator, target, opcode, qname, qtype, rcode, or a regular expression to be specified.

In designing Ncap the decision was made to no longer capture layer-2 network information such as Media Access Control (MAC) addresses because they weren't very useful and are generally considered sensitive information. Furthermore, Ncap defines file and stream formats for both input and output, and allows for multiple simultaneous inputs, which are both features that make the SIE possible.

The Ncap format, which is used both when broadcasting data streams and writing to files, adds little overhead to the DNS message. Each file begins with a simple header identifying the version of Ncap used and the file descriptor. Each DNS message is kept essentially intact, except that at the beginning of each DNS message two fields are prepended, termed simply "user 1" and "user 2." In practice, user 1 is used as a counter for how many times an RR set has been observed on the deduplicated broadcast channel. For example, if 3 copies of a packet are observed during the time window, only 1 copy is rebroadcast, however that copy will have a 3 in this field. The other field customarily contains the unique identifier assigned to the point at which the packet was collected.

As of fall 2009 development of Ncap has ceased and it is considered to be in its final form. Work has begun on its successor, Nmsg, and builds are available as of this writing from http://ftp.isc.org/isc/nmsg/. Nmsg is backwards compatible with Ncap, although it is not known when Nmsg will supplant use of Ncap. As of April 2010, the SIE data feed still runs on Ncap.

2.2 IMPROVEMENTS TO NCAPTOOL

There were both minor and significant improvements made to Ncaptool in order to facilitate this DNS research. All of the following improvements are included in the last version of Ncap that is available from the SIE. The more minor improvements included adding some RR types to those that Ncaptool recognized as well as some query types, and correcting the filter specifications so that the "initiator" and "target" fields were actually those machines and not merely synonyms for "source" and "destination."

However, the primary enhancement to Ncaptool was the development of the qnames.so plug-in module. This plug-in was necessary to provide the functionality of searching through a huge number of DNS messages for a long list of domain names. Ncaptool provides the functionality to search for one qname, or for a regular expression, however it does not provide for any way to search for multiple distinct domain names. The most computationally expensive step in this search is unzipping each ncap file and stepping through it to each DNS packet to compare its query name to the target, and that happens millions of times per search. Qnames.so allows the researcher to make only one pass through the target date range for the list of domain names, instead of 5000 passes. Even if the names in the list could be reduced to a regular expression, it takes seven hours for Ncaptool to search one week of messages for a single regular expression. Using qnames.so, it took one hour and a half to search the same number of messages for any of a list of 5300 domain names. Furthermore, qnames.so allows for the use of wildcards within subdomains. For example, within the text list one could specify *.evil.co.uk instead of trying to enumerate every subdomain of evil.co.uk in the list.

In addition to qnames.so a small utility program was developed to supplement the capabilities of Ncaptool. The utility, called ncapappend, is a stand alone program that is to be used in conjunction with Ncaptool when multiple ncap files are being unzipped. It takes as arguments what the input and output files are, '-' representing standard in or out, and removes ncap file headers besides the first one. Without this functionality multiple ncap files cannot be sent to Ncaptool for processes because it crashes when another file header comes in after the first one. A common implementation is 'gunzip *.ncap | ncapappend - - | ncaptool …'

2.3 COLLECTION AND ANALYSIS SCRIPTS

Several different bash scripts are used to gather and maintain the collection of Ncap files. Many of these are included in Appendix B for easy reuse; however note the license agreement at the beginning of the appendix. Several scripts were used to do the various pieces of analysis with the DNS data. These ranged from quite simple to things that should probably have been done in a compiled language. However, efficiency was not a primary goal of this project and neither was software engineering, and the analysis machinery was all written in bash and not ported to another language when it began to stretch the bounds of that approach.

The broadest and most simple scripts involve the capture and sorting of the Ncap files from the SIE data feed. The most basic of these scripts is MaintainDirectoryCreation.sh. It is to be run sometime before midnight in order to create the directory for the next day. A slightly more complicated script is

MaintainRollingCapture.sh, which serves two functions. First, it checks if the Ncaptool process is running, and if not, restarts it. Since this is a critical consistency check, this script is run every 5 to 10 minutes. Its second function is to move all of the Ncap files out of the temporary directory into which they are initially captured to a directory for the day during which the file was recorded. Ncaptool automatically writes files with a timestamp, and this convention is utilized in identifying what day's directory to which to send the file.

The third script is the most complicated one involved with maintenance of the Ncap files. This script, MaintainDiskSpace.sh, checks to see if the disk storage is overly full, and removes the oldest directory if the percentage of used disk space is above a certain threshold. It contains the added functionality of removing unique answer records from each day's DNS messages and storing them in a text file. This reduces the storage cost of each day from 17 GB a day to about 400-500 MB a day and continues to allow analysis of the DNS data. Particularly, these condensed daily records have allowed for a much more robust usage of the algorithms to determine if a domain has been newly registered. These algorithms are utilized in the discussion in section 3.4.

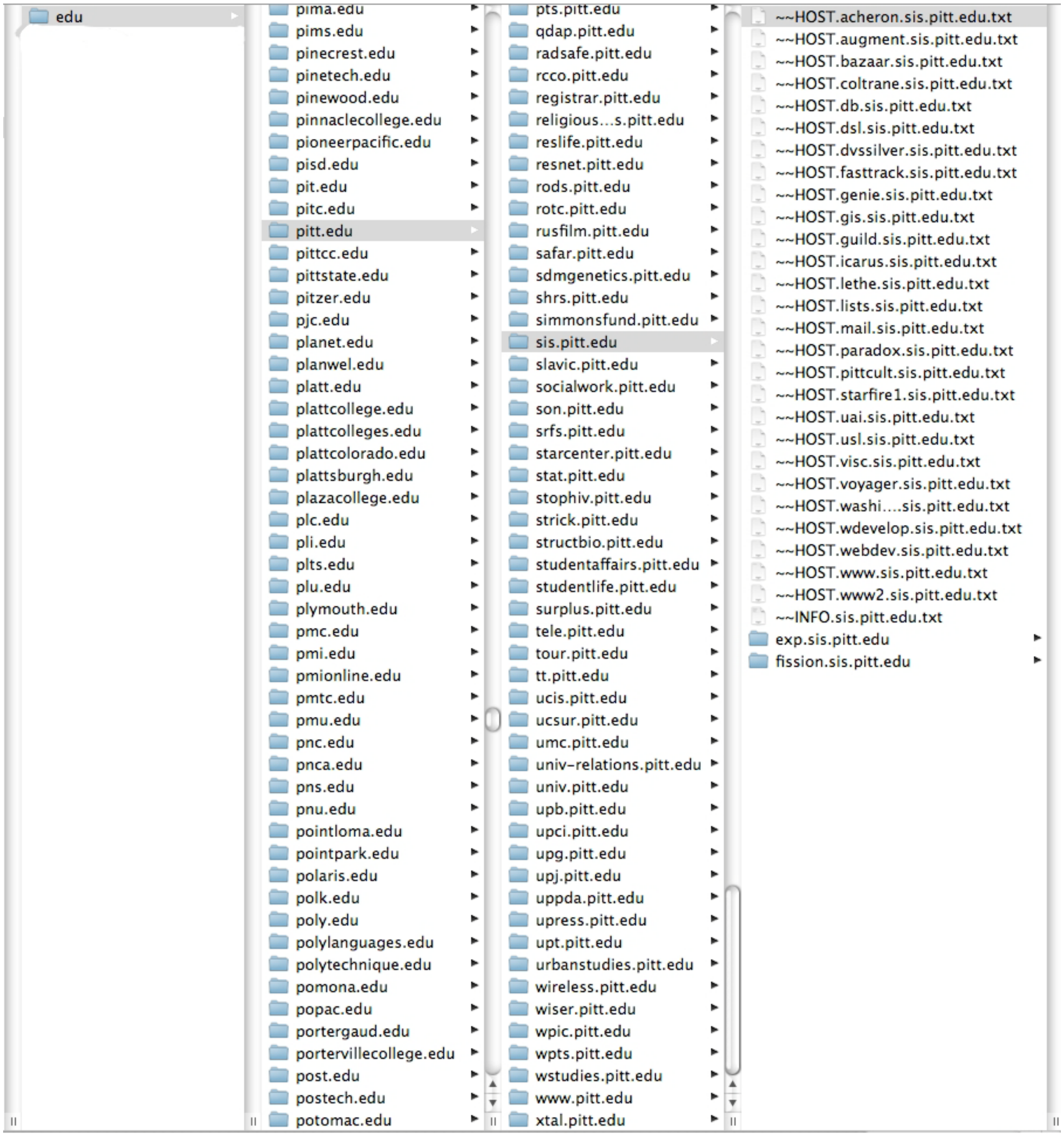
The other scripts in Appendix B were used for analysis of the DNS data. These fall into two sets of scripts for two different tasks. One set is for the mapping out of the hierarchical structure of a domain from the observed DNS data, and the other is to help shed light on the question of how much time it takes a phishing site to be put on an anti-phishing block-list. A detailed discussion of the logic in the scripts is not offered, rather a view of what they accomplish. For detail on the actual logic it is simplest to refer to the code and in-line comments in Appendix B.

The domain mapping scripts operate in two basic phases. The goal of the first script is to prepare a text file that will be usable by the second script to build the desired structure. The contents of this preparatory text file are selected on some basic premises. Firstly, it is observed that the information that is of interest to describing the interrelations in the target domain exists almost exclusively in the resource records of the DNS packets. As such, the actual DNS packets, and duplicate resource records, are not necessary. `Ncaptool`'s regular expression matching capability is utilized to narrow down the scope of DNS packets handled in text file format. Without this narrowing, the list is too large to be used in a reasonable amount of time. Regular expression matching is used instead of the `qnames.so` plug-in because this search is concerned with every resource record in the DNS packet, and not just the query, because any information relating to the domain in question adds to the completeness of the description. Once these packets are sorted out, `Ncaptool` is used again to convert them to text and then various algorithms are applied to standardize the form of the resource records in the text list. This standardization is necessary so that the list of records can be sorted and the unique records identified properly and kept in the list. In this state the list can be passed to the second phase of the process.

The second phase of the process utilizes the innate directory hierarchy structure of operating systems to sort out each resource record in the list. The list is fed into a loop, and for each record a file path is created for the name or names in the record. Any resource record that refers to that name is then added to a file in the appropriate directory. Additionally, any information about the domain generally, which usually includes such record types as those that involve DNS-SEC (DNS security functionalities) and start of authority (SOA) records are included. For example, a folder for the .edu top level domain would have an information file with anything that mentioned exactly '.edu' as well as several directories for subdomains within .edu such as pitt.edu. These directories would have the same structure until the directory had no further subsidiary directories.

The second phase of the domain map contains one other decision making process. The script attempts to determine if a domain name mentioned in a record is for a host or for a domain. This distinction in not made innately in the DNS protocol, and so some rough heuristics are applied to attempt to make the distinction. This distinction is helpful because if a name is a host, and not a domain, then the domain name does not have any subdomains, by definition of a host. For a human user, it would be helpful to represent a host as an endpoint in the directory structure and not as yet another folder. The endpoint analog in this case would be a file that contained all the resource records relevant to that host. This helps make clear what exists within a domain. The basic heuristic applied is that if the name has an answer or canonical name record about it, or it claims to be a name server for anything, then it is a host. Since these records can appear about domains also, the script then checks to see if there are any records indicative of being a domain and if so overrides the decision to call it a host.

The result of this sorting is a file hierarchy that can be viewed using any common GUI operating system. A sample of the output is shown in Figure 2. If this approach were to be taken on a large scale it is advisable to utilize a file format that does not require a minimum file size of 4 KB, however even for a domain the size of .edu a standard home PC or Mac is able to handle the directory structure fine. Figure 3 displays a sample of the contents of a host's file in this structure, which contains a line for each RR that was found to include the name of that host in some way.



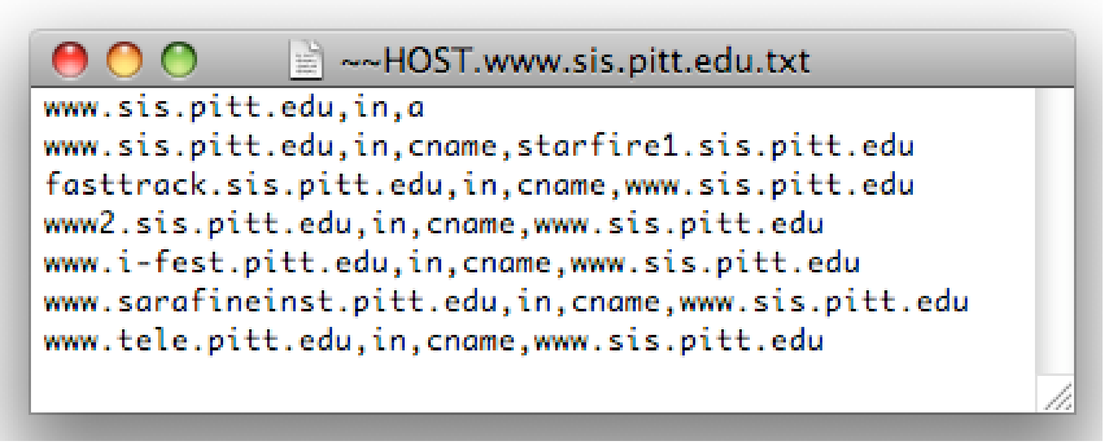
```
edu

pima.edu
pims.edu
pinecrest.edu
pinetech.edu
pinewood.edu
pinnaclecollege.edu
pioneerpacific.edu
pisd.edu
pit.edu
pitc.edu
pitt.edu
pittcc.edu
pittstate.edu
pitzer.edu
pjc.edu
planet.edu
planwel.edu
platt.edu
plattcollege.edu
plattcolleges.edu
plattcolorado.edu
plattsburgh.edu
plazacollege.edu
plc.edu
pli.edu
plts.edu
plu.edu
plymouth.edu
pmc.edu
pmi.edu
pmionline.edu
pmtc.edu
pmu.edu
pnc.edu
pnca.edu
pns.edu
pnu.edu
pointloma.edu
pointpark.edu
polaris.edu
polk.edu
poly.edu
polylanguages.edu
polytechnique.edu
pomona.edu
popac.edu
portergaud.edu
portervillecollege.edu
post.edu
postech.edu
potomac.edu

pts.pitt.edu
qdap.pitt.edu
radsafe.pitt.edu
rcco.pitt.edu
registrar.pitt.edu
religious...s.pitt.edu
reslife.pitt.edu
resnet.pitt.edu
rods.pitt.edu
rotc.pitt.edu
rusfilm.pitt.edu
safar.pitt.edu
sdmgenetics.pitt.edu
shrs.pitt.edu
simmonsfund.pitt.edu
sis.pitt.edu
slavic.pitt.edu
socialwork.pitt.edu
son.pitt.edu
srfs.pitt.edu
starcenter.pitt.edu
stat.pitt.edu
stophiv.pitt.edu
strick.pitt.edu
structbio.pitt.edu
studentaffairs.pitt.edu
studentlife.pitt.edu
surplus.pitt.edu
tele.pitt.edu
tour.pitt.edu
tt.pitt.edu
ucis.pitt.edu
ucsur.pitt.edu
umc.pitt.edu
univ-relations.pitt.edu
univ.pitt.edu
upb.pitt.edu
upci.pitt.edu
upg.pitt.edu
upj.pitt.edu
uppda.pitt.edu
upress.pitt.edu
upt.pitt.edu
urbanstudies.pitt.edu
wireless.pitt.edu
wiser.pitt.edu
wpic.pitt.edu
wpts.pitt.edu
wstudies.pitt.edu
www.pitt.edu
xtal.pitt.edu

~~HOST.acheron.sis.pitt.edu.txt
~~HOST.augment.sis.pitt.edu.txt
~~HOST.bazaar.sis.pitt.edu.txt
~~HOST.coltrane.sis.pitt.edu.txt
~~HOST.db.sis.pitt.edu.txt
~~HOST.dsl.sis.pitt.edu.txt
~~HOST.dvssilver.sis.pitt.edu.txt
~~HOST.fasttrack.sis.pitt.edu.txt
~~HOST.genie.sis.pitt.edu.txt
~~HOST.gis.sis.pitt.edu.txt
~~HOST.guild.sis.pitt.edu.txt
~~HOST.icarus.sis.pitt.edu.txt
~~HOST.lethe.sis.pitt.edu.txt
~~HOST.lists.sis.pitt.edu.txt
~~HOST.mail.sis.pitt.edu.txt
~~HOST.paradox.sis.pitt.edu.txt
~~HOST.pittcult.sis.pitt.edu.txt
~~HOST.starfire1.sis.pitt.edu.txt
~~HOST.uai.sis.pitt.edu.txt
~~HOST.usl.sis.pitt.edu.txt
~~HOST.visc.sis.pitt.edu.txt
~~HOST.voyager.sis.pitt.edu.txt
~~HOST.washi....sis.pitt.edu.txt
~~HOST.wdevelop.sis.pitt.edu.txt
~~HOST.webdev.sis.pitt.edu.txt
~~HOST.www.sis.pitt.edu.txt
~~HOST.www2.sis.pitt.edu.txt
~~INFO.sis.pitt.edu.txt
exp.sis.pitt.edu
fission.sis.pitt.edu
```

Figure 2: Example domain mapping output



```
www.sis.pitt.edu,in,a
www.sis.pitt.edu,in,cname,starfire1.sis.pitt.edu
fasttrack.sis.pitt.edu,in,cname,www.sis.pitt.edu
www2.sis.pitt.edu,in,cname,www.sis.pitt.edu
www.i-fest.pitt.edu,in,cname,www.sis.pitt.edu
www.sarafineinst.pitt.edu,in,cname,www.sis.pitt.edu
www.tele.pitt.edu,in,cname,www.sis.pitt.edu
```

Figure 3: Example contents of file in domain mapping output

Whereas the domain mapping scripts produce a nearly finished product on their own, the scripts used for the analysis of the lifecycle of phishing sites are primarily a means to acquire the statistics that will then be analyzed. Furthermore, several of the phishing analysis scripts involved standardizing inputs into the script that managed the search of the Ncap files. These standardization scripts are not included because they accomplish mere text formatting that is specific to the source of our phishing site block list.

The search script, `manySiteSearch.sh`, is included in Appendix B. It takes as inputs a text file that is a list of domain names as well as start and stop dates for the search and outputs all DNS packets from the data source whose query name matches any of a list of domain names into an Ncap file. The next script included in Appendix B, checkEarliestDates.sh, takes this ncap file and the text list, as long as the text list included dates for when the domains were added to it, and calculates the difference between the date on the list and the date of the first DNS packet that asks for the domain name in question. These differences are the basic data for the analysis of how long it takes a phishing site to be added to a block list, but as will be explained in section 3.2, there are several other factors in between the list of differences output at this stage and the complete answer to the question. The third script included in this section, `evaluateDNhack.sh`, aims to discern whether or not a given phishing site was hosted on a hacked server, thus hijacking that server's domain name for malicious purposes, or if the domain name was registered exclusively for the purpose of hosting malicious content. It organizes these answers into a basic histogram, counting the sites separately in different categories based on the latency between when the site was in the DNS traffic and when it was block-listed. The script determines this by querying for the site, and keying on the HTTP return code. If the lookup is successful, then it is assumed that the site has been cleaned and reopened, whereas if the return code is an error or redirection it is counted as having been hacked because no one cared to clean up the site and bring it back online after it was blocked. Redirections are counted here because it was observed that ISP or other intermediaries often do not simply deny requests for known malicious sites, but redirect the user to a page that explains the site that they requested is a known malicious site and gently admonishes them to be more careful.

3.0 ANALYSIS

DNS data is a rich source of information about many of the events and transactions of the Internet. Therefore, there are several different applications of the data collected in this research. This chapter presents the several approaches that were investigated in the available time, however it is not an exhaustive list of the types of analysis one could conduct using DNS data. It is an interesting and fruitful protocol to research for several reasons, as highlighted in section 1.1. The applications and interpretations of the DNS data will hopefully continue to grow and change as more researches take interest in the protocol's value. This analysis begins that path.

3.1 DOMAIN MAPPING

The attempts at domain mapping were undertaken simply to see what could be passively constructed with a comprehensive DNS source such as the SIE data feed. The original hypothesis, which was hardly discussed, was that it wouldn't turn up anything that interesting or that impressive. However, the file structure that was built up from the DNS data using some primitive scripts was quite comprehensive. By sampling about two weeks of DNS traffic a complete-looking description of whole top-level domains could be generated in a very naturally explorable format (see figures 2 and 3 for sample output). For smaller top level domains (TLD) the structure was generated at a speed that the files could be updated daily without falling behind the production of data.

The information that is included in the generated description includes some potentially sensitive information, however none of it is data that is not public to anyone who cares to ask. IP addresses will be revealed to the world if you publish answer records. Not only logical locations of machines are revealed through DNS, however, because records such as cname records reveal what services are probably co-located on one machine. Start of authority records can indicate what machine or organization administrates for other machines or organizations and therefore describes possible trusted relationships. There were also a few organizations, which shall remain nameless, whose structures were unorthodox in myriad ways, and this sort of information can be used to make inferences about the quality of technical oversight. These indicate the most obvious sensitive inferences to glean from the maps; however there seem to be many more bits of information one could extract with a fine-toothed comb and some perseverance. It is this extra information that differentiates this approach from previous domain mapping attempts, as these approaches incorporated only IP addresses and domain names into their maps.

All of these tidbits of information are items that in a secure environment one would prefer not to share with the world. At the very least, the operator of a domain should want to know what information about the domain is publically available if privacy is a concern for the domain at all. Operators could also check this data for inconsistencies or anomalies, and certain anomalies may be more easily discoverable in this data. For example, if your name server were claiming to be authoritative over an unknown domain, it would be clearly displayed in the file for the name server, even if the frequency were very low.

There are further curiosities that arise when using these scripts in creative ways. Through chance, it was noticed that McAfee.com was claiming certain other domains as a subdomain. For example, entries such as ‘www.sis.pitt.edu.phishing.mcafee.com’ were noticed. To investigate, the scripts were pointed at mcafee.com as the top-level domain to search under. This search returned a very interesting domain tree, which included a couple of different subdomains which in turn included fully qualified domain names from all over the internet such as the example above. It seems that these domain names are registered by the company as a way to communicate the danger of sites, as different response codes to the DNS lookup of these domain names at McAfee indicated different levels of trust in the safeness of the indicated domain. This appears to be simply an instance of intentional misuse of the DNS database for other purposes, such as those discussed by Paul Vixie recently (Vixie 2009). Another site that was noticed to operate on a very similar principle was rfc-ignorant.org.

Both the McAfee and Rfc-ignorant data can be leveraged in interesting ways as well. It can be used to identify what sites are thought to be malicious in the one case, or which sites are reported to violate various RFCs in the other. When viewing large swaths of the domain space it is interesting to use this data to pick out what domains fall into these categories, and this is one of many sources of data about a domain that an administrator might like to monitor for unusual activity about her network.

The amount of information gleaned is particularly interesting because it is essentially public. Anyone with a capability to listen to enough Internet traffic should be able to acquire such a map of any domain that communicates directly with the Internet. It is unknown how many collection points and their necessary data rates in order to construct a comprehensive DNS mapping such as was obtained here. However in theory all DNS data is public and the whole worldwide DNS database is connected, so persistent, small actors could acquire this amount of information.

Regardless, continuing with the collected ".edu" example, we observed 6,537 second level domains. Educause, the official registrar for the domain, reports that there are about 7,000 second level domains (McTurk 2010). So merely by listening to public DNS requests we acquired information about 93-94% of the second level domains. It is unknown the popularity or size of the domains that were missed, but intuitively the smaller, less requested sites are the ones absent from the data. Following this logic, if the percentage of records included down through the second and third level domains could be calculated, it is predicted this percentage would be approximately equal. The percentage is difficult to calculate because many of these 6,500 institutions would need to be contacted to check the number of domains they have, however a cursory inspection of the lower level domains does not indicate any obvious gaps in information. In order to increase completeness a larger sample time can be queried from the ncap database, however this requires a constant increase in the time to build the domain structure and provides diminishing returns insofar as percentage of records included.

3.2 PHISHING RESPONSE TIME

Phishing attacks involve malicious web pages that attempt to trick users into divulging sensitive information such as social security numbers, credit card numbers, or user names and passwords. These attacks can cause a massive amount of financial damage to both the targeted trusted entity, such as a bank, as well as the entity's customers. Figure 4 presents a graph of the estimated cost of a particular phishing attack over time, reaching a total of about $4.3 million over 72 hours (Cyveillance 2008).

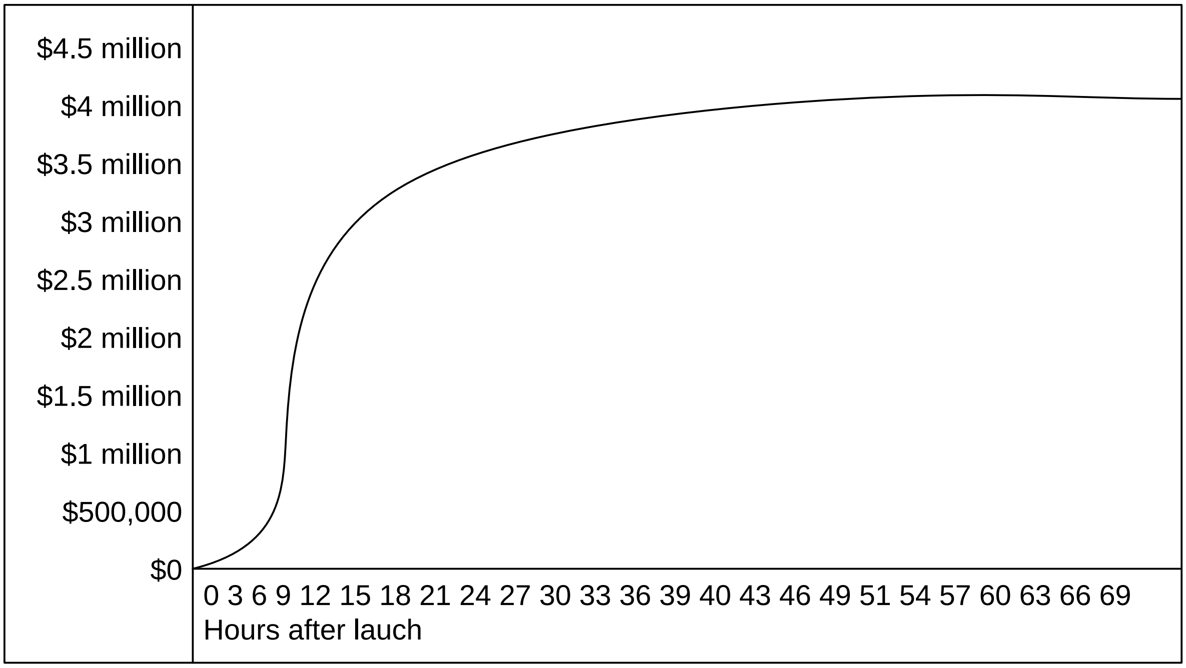


Figure 4: Approximate cost (USD) vs. Duration of attack (hrs) for an example phishing attack (Cyveillance 2008)

As shown in the figure, phishing attacks often cause the majority of their monetary damage early in their deployment. Figures for the number of users compromised take a similar shape. There are several parties that take interest in minimizing the damage done by phishing. It has been demonstrated that publishing "block lists" or "black lists" for consumers and, in some cases, Internet service providers (ISP), causes a significant reduction in the time that a phishing site can remain active (Moore 2007). There is also good information as to how long phishing sites stay active (Aaron 2009). The goal of this part of the research was to shed some light on the latency between initiation of a phishing attack and the phishing site being listed on an industry block list. The analysis of DNS traffic was accomplished by examining records collected from the Internet Systems Consortium Security Information Exchange (ISC-SIE). The ISC SIE is a trusted, private framework for information sharing in the Internet Security field.

The basic premise is simple. A user's computer would have to ask where the phishing site is before the user could be taken there to be tricked into revealing information. This timestamp can then be compared to that on the block-list and the difference is how much time elapsed between the site going live and being put on the block-list. Given access to the SIE data, we could look for the first instance of when the name was requested in the DNS traffic and have a fairly accurate measure of when the site began receiving traffic.

The block-list utilized in this research was downloaded daily and checked, even though the block-list is updated more frequently. Its provider has requested that it remain anonymous. Over the course of 135 days the list contained an average of 3603 unique URL's per day, with a median number of URL's equal to 2324. The standard deviation was 2860 URL's, with the list skewed to the right. The number of unique domain names was significantly lower, and the date-added-to-the-list was not valid a date for several entries. For example, the date was in the future, or 20 years ago. Normalizing the list to acquire a list of unique domain names with valid dates reduced the size of the list to an average of 2002 entries per day, with a standard deviation of 1266, and reduced the median number of entries to 1550. During the whole period approximately 101,000 unique phishing sites were reported, for an average of 581 new unique sites per day.

3.2.1 Difficulties in analyzing DNS data from the SIE data feed

There are several potential limitations in using DNS data for this analysis. For example, the assumption that a phishing site's launch can be detected by its first appearance in the DNS traffic is not always true. If a phishing site utilizes static IP addresses instead of domain names no DNS request will be issued. Luckily for DNS analysis, this style of phishing represents 8.4% of all phishing addresses observed in the second half of 2008 and 10.5% of those in the first half of 2009 (Aaron 2009; Aaron 2009). Attacks using these phishing sites will be invisible to the DNS analysis technique. None-the-less, being able to evaluate the performance of the block-list in finding roughly 90% of phishing sites is still significant.

Another limitation arises from the fact that in order to operate efficiently, the DNS protocol makes extensive use of caching and a distributed infrastructure. So if a server receives an answer to a question, the server will store the answer for a period of time so it can respond itself instead of referring the question to another server. Furthermore, UDP DNS messages between hosts are not the only way of transferring DNS information. DNS servers can periodically initiate zone transfers between themselves, if they are so configured, to update their records en masse. The SIE data would not reflect these zone transfers. The problem with these two features of DNS is that the SIE sensor points are irregularly distributed throughout the Internet in locations that are not allowed to be disclosed. There is thus some uncertain probability of information about a phishing site eluding our sensors. However, there is reason to believe that the information obtained through SIE is representative of the Internet as a whole. Anecdotally, the ISC SIE has a reputation for reliability, and a high degree of trust that they have acquired high-profile and quality sensor points. Verification of complete coverage may be found in the completeness of the domain maps constructed and described in section 3.1, which were found to contain just over 93% of the second-level domains that are registered in the ".edu" top-level domain.

Most vexing for this research is that according to the most recent statistics made available by the Anti-Phishing Working Group (APWG), which are for the first half of 2009, only 14.5% of phishing site domains are actively registered by the phishers; this is down from 18% in the second half of 2008 (Aaron 2009). This is extremely problematic for the proposed analysis method using DNS because virtually all of the rest of the domain names used for phishing were comprised of hacked domains that already existed in the DNS traffic before they were compromised. These packets completely invalidate the DNS packets as a timestamp of when the malicious site became active and started attracting targets. This situation means approximately 15% of the sites that are observable are valid data points for analysis. Further complicating the issue, the bad data points are not merely undetectable; they introduce a high degree of noise into measurements that can be made.

The lists provided other questionable data points as well, regardless of whether the domain was maliciously registered or hacked. The problem with these data points centers around the observation that several of the entries on the block-list cannot be considered to be wholly independent events. For example, consider the six entries in Table 1. There are two distinct confounding factors represented by these entries, which are almost certainly maliciously registered domain names and so otherwise valid data points. First, two of the entries are for sites with identical domain names (those numbered 244). This occurs because the block-list uses hypertext transfer protocol (http) addresses, because multiple websites can be hosted on one domain name. This allows the block list to be more fine-grained and not block entire domains if not necessary. This is particularly beneficial when there are innocent bystanders – websites that are not malicious on domains that have been hacked. However this feature can skew the data because every domain name will have the same first observed date while different dates they were block-listed. This can lead to a phenomenon in which, as far as this DNS data is concerned, the domain is blocked before it is requested. But even if this is not the case, it appears to give the block-list organization a head start on finding the site, because another site of the same domain name already existed. To ameliorate this issue, only unique entries in the block list are kept, and the earliest time stamp for a given unique name is kept.

Table 1: Example Dependent Block-list Entries.

Domain name	Date block-listed	Date first observed	Difference
businessconnect.comerica.com.session-id-**233**.fikhi.com.mx	2009-07-14 17:46:56	2009-07-14 15:01:23	0000-00-00 02:45:33
businessconnect.comerica.com.session-id-**244**.fikhi.com.mx	2009-07-14 14:17:57	2009-07-14 15:28:20	^No requests before phishing Listing
businessconnect.comerica.com.session-id-**244**.fikhi.com.mx	2009-07-14 15:57:47	2009-07-14 15:28:20	0000-00-00 00:29:27
businessconnect.comerica.com.session-id-**246**.fikhi.com.mx	2009-07-14 16:34:40	2009-07-14 18:35:46	^No requests before phishing Listing
businessconnect.comerica.com.session-id-**257**.fikhi.com.mx	2009-07-14 17:44:23	2009-07-14 17:00:13	0000-00-00 00:44:10
businessconnect.comerica.com.session-id-**270**.fikhi.com.mx	2009-07-14 18:00:26	2009-07-14 17:35:54	0000-00-00 00:24:32

The other issue indicated by the entries in Table 1 is that all six clearly represent a correlated string of registered domain names. If an employee of the block-list organization notices a run of site names like this, the sensible thing to do would be to block some broader swath of the domain space. The block-list organization does employ humans for this sort of task. A confidence rating of 100% is only provided after a human has verified an entry. This issue is more difficult to mitigate than the previous issue. Here, it is a legitimate foresight by the blocking organization to do this and does not unfairly bias each individual data point. However it is also often an unduly large percentage of the total domain names on a given list that are registered in this way due to the volume of domains registered en masse. The importance of discovering this one massive registration event over other single registration events is inflated, even though there is no evidence as to which actually caused more harm and was more important to find. These incidents are also much more difficult to cleanly filter out.

3.2.2 Countermeasures to increase data validity

Two steps were taken to reduce problems that might be caused because of questionable data. The first was to throw away entries for which the difference between the date blocked and the date observed in the DNS data was negative; i.e. the blocked date was before the observed date. This action allows for several of the sites to remain in the data, while eliminating several of the packets that could inflate the data or result in an average of negative value, which is a nonsensical answer to the question of how long it takes to find a phishing site.

The second step was to statistically weight data points based on how quickly the domains were discovered. Since many of the data points on our list were unusable because the domain name in question existed in the DNS traffic long before the site became a phishing site, a complicated schema had to be devised to enact a simple premise. Basically the longer the latency between a site being blocked and the beginning of the site's DNS activity, the more likely it is to be a hacked site. This is because, hypothetically, if www.harmless.com has existed since 1998, and it was hacked and blocked in 2009, it should exist all the way back to the edge of our database. Chance comes into play in that it is possible that a DNS request for www.harmless.com was not observed by the SIE sensors for 15 days, and then one occurred, and then nothing, and then 15 days later when it was hacked a flurry of activity happens just before it is blocked. It will appear in the data with a latency of 15 days. Principally, the probability that a site was a hacked site, and thus an invalid data point, increases proportionally with the value of the difference between the date observed and the date blocked. The difficulty lies in discovering exactly what proportion that is.

The relationship between latency and chance-hacked was verified by hand. The results were sorted by the difference in dates and each site was judged by the human observer as to whether or not they appeared to have been hacked or maliciously registered. Clues were taken from various locations. The pronouncibility and simplicity of the domain name was taken into account, as was its perceived deceitfulness. Each site was also visited. Through this process it became clear that many domain names that were clearly registered maliciously were no longer accessible, with various error messages being presented. These error messages ranged from simple “page not found (404)” errors to redirections to ISP pages warning the user that the site they attempted to visit was a malicious site. On the other hand, hacked domains had often been cleaned up and were accessible again and their forward facing pages appeared legitimate.

This observation led to an approach to automate this evaluation process for the large numbers of domain names on each of several lists that needed to be counted so a probability could be calculated. The strategy is that if sufficient data points are collected about the change in probability of being a maliciously registered site versus listing latency, then a curve can be calculated to fit those data points. This regression function can then be used to weight the validity of each data point based on what its latency value is. Thereby, if there is a 90% chance that a given data point is valid based on its short latency value, say 2 hours, and another data point, one whose latency is much longer, say 9 days, has a 10% chance of being a maliciously registered data point; the point with a short latency will count towards the average latency of all sites blocked nine times as much as the other. This is because it is much more likely to have been maliciously registered and therefore a data point that we can accurately measure in DNS data.

The script `evaluateDNhack.sh` contains this automation attempt. By keying on the http return code when attempting to access the website, the script is able to count whether or not each site on the block-list has been taken down or reopened. The percentages are calculated for blocks of time, using 4-hour windows for latencies of less than one day and daylong windows for greater latencies. The window size was increased because there are insufficient phishing sites still undiscovered after one day for the smaller window to produce valid sample sizes. The results from this method correspond with the results acquired from by hand, however at low and high latency values it varies noticeably from the human estimations. However, since it is not known if the human estimation is actually correct in the first place, it is not clear how to handle this phenomenon. Primarily, the data was adjusted by having a cut-off value, after which the site in question was assumed to be hacked and an invalid data point. This cut-off was after a full 14-day latency, and was based on the manual observation of sites. The probability that any phishing site is alive for this long is scant, around 0.2% (Moore 2007). Therefore, the chances that a request two weeks before the domain was blocked is due to malicious activity is close enough to zero that, in conjunction with the observed evidence most sites at this high of a value appear to have been hacked, that the data is cut off and a value of 0 is manually input for the trustworthiness of day 14.

Data was collected over two distinct intervals. The first was eight days of collections for eight block-lists from the end of July and first week of August 2009, which will be referred to as interval 1. The second interval comprised of 28 days between February 20th and March 23rd, 2010. These timeframes were driven primarily by external factors, such as availability of computing resources or internship and school schedules, and were not selected for any experimental design purpose.

3.2.3 Results

The two distinct intervals of data collection are sufficiently different to deserve individual presentation and discussion. Within interval 1, collected late July and early August 2009, the statistics varied widely from block-list release to block-list release, however they appear to be evenly distributed. During interval 1 the median values of the latency between first observed DNS request and the time of blocking per day's list ranged between 3:51 (HH:MM) and 16:20. The average of the medians is 11:02, while the median of the medians is 11:18. The average latency ranged between 31:52 and 54:37 per each day's block-list. The average of the averages is 41:58, while the median of the averages is 41:32. Since the average and median values of both of these two ranges are nearly the same, the values should be nearly normally distributed.

These values are based on an estimation of trustworthiness of whether or not a site is a valid data point. This estimation is used to weight each latency value before it is included in the average and median values. The regression equation that was fitted to this trustworthiness estimation was the following: $P = [(1-t^{1.542294}) / (3585.228*t)] + 0.699$

Where P is the probability that a site is a valid data point and t is the time, or latency, in seconds between when the site occurs in DNS data and when it was block-listed. The r-squared value for the equation is .90. This equation is dynamically applied to each t value to weight it when calculating both the mean and the average for the above values. Table two contains a summary of the data that was collected during interval 1. Note the large number of the sites that were found and block-listed within about 4 hours of their going live.

Table 2: August phishing entry totals

Latency is greater than (days)	Total suspected Registered (automated)	Total Phishing Entries	Percentage
0.0	692	949	72.92%
0.21	74	120	61.67%
0.42	77	104	74.04%
0.63	43	66	65.15%
0.83	76	92	82.61%
1.0	125	200	62.50%
2.0	58	115	50.43%
3.0	34	85	40.00%
4.0	34	81	41.98%
5.0	24	71	33.80%
6.0	17	63	26.98%
7.0	41	77	53.25%
8.0	28	66	42.42%
9.0	33	69	47.83%
10.0	20	62	32.26%
11.0	11	86	12.79%
12.0	24	81	29.63%
13.0	19	78	24.36%
14.0	22	80	27.50%
15.0	30	120	25.00%
16.0	25	120	20.83%

Table 3 compares the average and median values for the lifetime of phishing sites that are reported in APWG’s biannual report on the status of phishing on the Internet to those that were acquired for the interval 1 data. While the information on the standard deviation size for the APWG reports is not available in the report, it can be calculated for the observed latency data. The standard deviation for the median latency is 5 hours, zero minutes, and for the average latency it is 7 hours 35 minutes, which indicates that the data is widely variable. For reasons explained in the following, more data did not alleviate this high variability. However it is clear from the data that both of the reported medians fall within one standard deviation of the median time it takes a site to be block-listed. Taking the values as is, which must be done cautiously due to the high variability, one would conclude from the medians that it only takes 2-3 hours for a site to be shut down after it is put on the block list. The rest of the time, 11 hours, is the time it takes a site to be listed after it begins phishing.

Table 3: Phishing lifetime statistics compared to anti-phishing response time

	APWG 2H2008 phishing lifetimes	APWG 1H2009 phishing lifetimes	Observed latency to list a phishing site (August 2009)
Average value (HH:MM)	52:02	39:11	41:45
Median value (HH:MM)	14:43	13:16	11:02

Let us return briefly to figure 4, and compare the time value of the second inflection point in the graph to the median lifetime values reported by APWG. The point at which the damage done by the site significantly abates is between 12 and 14 hours after launch, at which time this particular attack had already caused some $3 million in damages. In just two hours between 10 and 12 hours after launch the attack did approximately $800,000 in damage. It is obvious that the return on investment of reducing phishing times by even an hour or two is great, as just a couple hours can save a tremendous about of money for victims. As phishing becomes increasingly profitable for malicious actors and more resources are devoted to separating web users from their money, the Internet community must take steps to safeguard the legitimacy of the web space. Even though returns may appear to be small in terms of reducing uptimes, phishing attacks deal with a tiny profit window and any reduction of that prime revenue generating window is of great importance.

The interval 2 data was collected with the intent of corroborating the statistics from interval 1. However within interval 2, collected late February through late March 2010, the data is problematic to analyze in the same manner at all. The reason for this can be apprehended readily by considering table 4, which presents the total number of phishing entries and the number suspected to be maliciously registered ordered by their latency of discovery.

Table 4: February and March phishing entry totals

Latency is greater than (days)	Total suspected registered	Total Phishing Entries	Percentage
0.0	162	271	59.78%
0.21	108	242	44.63%
0.42	73	199	36.68%
0.63	81	203	39.90%
0.83	47	121	38.84%
1.0	203	629	32.27%
2.0	196	669	29.30%
3.0	136	544	25.00%
4.0	128	553	23.15%
5.0	122	700	17.43%
6.0	122	722	16.90%
7.0	138	791	17.45%
8.0	177	1070	16.54%
9.0	204	1369	14.90%
10.0	405	2246	18.03%
11.0	699	4031	17.34%
12.0	636	3549	17.92%
13.0	468	2386	19.61%

When contrasting table 2 with table 4 one difference is immediately apparent. The number of phishing entries reported in table 4 is extremely bottom heavy; the distribution of sites versus the latency of their discovery is reversed from table 2. This is problematic for analysis of the sort done with the interval 1 data because the majority of the noise, i.e. hacked sites that have been used for phishing, exists at the higher latency values. The method used during interval 1 to remove this noise is not sufficiently precise to rectify data in which there is up to 20 times as much noise as there is data; it operated with the noise in the reciprocal proportion.

The percentage of sites suspected to be maliciously registered via the automated tool is, on average, 25% lower within interval 2. The average and median latency values are completely suspect for this collection period because of the above reasons. As explained in section 3.2.1, the premise of the research is that sites that were maliciously registered will become active in the DNS data shortly, within hours or a couple days the majority of the time, before being blocked. Since according to APWG statistics 14.5-18% of phishing sites are malicious registrations, this should apply to approximately 15% of the data; the remainder of sites are hacked or otherwise are unanalyzable. In August, it was tenable to filter out these sites because 1646 sites were listed within 2 days while only 1139 took 3-16 days to list. There were many sites that were found simply at the edge of the database's collection limit, which obviously fell into the hacked category. So while these 1646 sites make up the majority, nearly 60%, of sites in question they do comprise about 15%-20% of the total number of domains on all the lists. The remainder 1139 that are still in question as viable data points can be sufficiently remediated to count the few important sites that would slip through the cracks if the analysis were just cut off at 2 days, which appears to be too heavy handed a tactic to generate useful results. For interval 2, which surveyed over 3 times as many days as the first interval, 2334 sites were listed within 2 days of going live and 17961 took 3-14 days. Here, only about 11% of the sites fall within the range that used to contain 60%. What accounts for this large swing in only six months time is an interesting question in its own right, however the data cannot serve its intended purpose of providing additional robustness to the data from interval 1.

3.3 FAST-FLUX DETECTION

A fast-flux implementation of DNS aims to have one fully-qualified domain name be directed to a multiplicity of IP addresses. One domain name could have thousands of IP addresses. Legitimate web servers pioneered the technique because it facilitates high availability and efficient load sharing. These are also traits that botnets and malicious web hosting strive for, and so it has been adopted in the malicious realm as well (Riden 2008).

There is basic, shared network architecture among these fast-flux networks, comprised of the compromised hosts, the backend servers, and the fast-flux "motherships." The infected hosts, which make up the botnet controlled by the phisher or other evil-doer, primarily serve as proxies that obfuscate the actual location of the backend servers (Riden 2008). It is these servers that actually contain the malicious web pages and will store any stolen data; the ever-changing IP addresses used within the botnet serve the traditional benefits mentioned above as well as providing those privacy and secrecy benefits of operating behind an anonymizing proxy service.

Even though fast-flux need not be a malicious activity, there are certain activities and attributes that are identifiable within DNS traffic that allow for a very high rate of successful automated identification of malicious fast-flux sites. Initially sites with 30 or more distinct A records in a single day are extracted from the data. These are further filtered for sites that have a low time-to-live (TTL) value. The domain name is then checked for the number of unique characters within the name. A greater number of unique characters is considered more suspicious because it has been observed that malicious sites tend to use a wider variety of characters. The remaining domain names are checked against the database of all domain names that have been observed on the data feed previously, dating back to mid-July 2009. Newly registered and little used domains are considered more suspicious, and this controls for known dynamic hosting services such as Akamai, which are observed daily. The sites remaining after this filter are newly registered fast-flux hosting domains, and they are weighted further by checking known sources for identifying malicious or benign domains. If the site in question is found upon querying Google News or some such service to be a legitimate site that has been newly registered to provide services it is heavily weighted towards being benign, whereas if it appears on an anti-phishing block list then it is confirmed as malicious.

The average number of fast-flux domain names detected over 174 days from September 4, 2009 to February 24, 2010 on the SIE data feed using these criteria was 64.5 sites. The median number was 58. However, the standard deviation was 59, and the data exhibits a long tail to the right, with a handful of extremely high values, the greatest of which was observed on February 17, 2010 at 525 unique domain names. The sum of every day's domain names yields a total of 11,222 observed domains; of these, 7,729 domain names were unique. Of these, only one is known to be a falsely identified malicious site that was actually benign.

Since fast-flux hosting is not specific to phishing, but could host any sort of malicious content, it would be expected for the list of detected fast-flux sites not to all be on the phishing lists. Likewise, since a phishing site can be hosted without the use of fast-flux hosting, it would also be expected that the fast-flux list would not be sufficient for detecting all phishing sites. During the 174 days summarized above, 2,962 unique sites were both detected by the fast-flux detection algorithm and listed on the phishing list. The phishing list contained approximately 101,000 unique domains over this period. The overlap therefore amounts to 2.9% of the phishing list, while 38.3% of the fast-flux sites are accounted for in this overlap.

However, for the time frame of February 20, 2010 to March 23, 2010 there were only 20 unique domains in common between the phishing list and the fast-flux detection algorithm. The reason for this is unknown. These dates are the same as those for interval 2 of the phishing block list response time study, which were also unorthodox. However it is unknown at what date between August and February these results became unorthodox, as the requisite data analysis was not done continuously, in fact not done at all, during the period that the fast-flux detection was run from September until February 20th. Therefore it is unknown if these observations are related in any way.

3.4 DDoS AND OTHER DISTRIBUTED MALWARE ANALYSES

The hypothesis with DNS detection of Distributed Denial of Service (DDoS) attacks is that if infected hosts are attempting to overwhelm a specific target, they will first have to figure out where that target is. This spike in requests should then be visible on the SIE data feed. This oversimplifies several difficulties in performing the analysis related to the DNS protocol and the SIE data feed. These problems were discussed in section 3.2.1 in relation to phishing response time analysis, however many of the same issues arise in regards to DDoS detection. IP addresses can be used as targets instead of domains, however statistics for how often this occurs were not found. Furthermore, when counting how many hosts are involved in an attack, the DNS protocol’s proclivity for caching responses becomes much more troublesome. Likewise, the uncertain distribution and completeness of the data feed is further exacerbated by the goals of DDoS analysis.

In addition to the relevant difficulties explained in 3.2.1, DDoS detection suffers from a problem of scale. It is not computationally feasible at this time to keep the status of all domains in order to notice when a spike of DNS requests about that domain occurs. If the SIE feed is being used to monitor for attacks on one's own network, it is computationally feasible to monitor yourself, however you would almost certainly have noticed internally that you were being attacked by the time the SIE data could tell you that. This is especially true due to the fact that DNS packets contain a variable time-to-live (TTL) field that describes how long a DNS answer should remain stored in a cache. If the attacked site has a long TTL, the chances that a DNS server outside of the stream of data collected by the SIE will already have the answer stored increases, thereby decreasing the likelihood it will be detected by the SIE. A low TTL for the target site's DNS information is observed to be correlated with the size of the spike in DNS requests spawned by a DDoS attack on the site. The lower the TTL, the larger the change in number of DNS requests is observed to be. This is yet another variable that would have to be controlled for in analyzing the size of a DDoS attack on a site or number of infected hosts involved in other phenomena.

For these reasons, the attempt at detecting DDoS attacks using the DNS data was not nearly so successful as the other analysis attempts made utilizing the SIE data. Even though DNS data is not a viable means for detection of these attacks, it is an additional source of information that one can utilize for a posteriori analysis of various widely-distributed phenomena. In DDoS attacks, the number of DNS requests for the targeted system could be viewed as a lower bound on the number of infected hosts involved in the attack. The data can also be valuable in describing the minimum number of infected hosts contacting malicious hosts that are used as command and control centers if these malicious hosts become known. These are lower bounds because even if one infected host makes several requests, those requests should also be cached on the first server it asks, which should not enter the traffic flow monitored by the SIE data. Similarly, it is likely to be an underestimation, as other infected hosts utilizing the same DNS server will receive the answer cached on that local DNS server, and so the request will likewise not be captured. One example where this was useful was in estimating the extent of Conficker infections in 2009 (Kriegisch 2009).

For these malware analyses it is unfortunately not viable to attempt to identify infected hosts using DNS data. Even though a DNS packet contains the IP address of the host making the request, this is often a recursive DNS server who is merely doing as it is asked and making the request on behalf of the infected host. Given all of these restrictions, it is obvious that DNS data would be playing an assisting role in analyzing malicious activity on the Internet. However, this particular analysis was also significantly hampered by the design goals of the data feed that was used; if a DNS data collection is designed with malware analysis as the goal the potential of DNS information should be reevaluated before it written off.

4.0 FUTURE WORK

It was desirable to utilize the DNS data to also confirm other measurements of the lifetime of phishing sites, especially because this data set would have allowed the correlation between when each site was blacklisted and how long after that time it was no longer available. However, the SIE data feed is not configured to allow such analysis because it only collects valid DNS messages, and the key message to determine the takedown of a site would be a “no such domain” message. This type of message is considered an error by the SIE data feed, for good reason concerning their primary goal of actively mapping a domain as it actually is. One aspect of future work regarding DNS traffic analysis would be to establish a data feed that is tailored to the needs of the analysis. However this is a large undertaking, and not something near the scope of this thesis. The work that is in the planning stages mostly involves refining the tools used in this analysis to languages or data structures better suited to the task at hand. Undertakings that might be more interesting are the two following ideas, which propose extensions of the work described in chapter 3 that would provide further insight into those questions.

4.1 AUTOMATED BLOCK-LIST GENERATION

It is hoped that the fast-flux detection algorithms could be used in conjunction with other resources to generate an automated block-list of malicious sites. Do to the high rate of publication of malicious sites and the extremely high value in blocking them as quickly as possible accurate and fast generation of an accurate and comprehensive block list would help increase Internet browsing safety. Other resources that would be leveraged into this attempt would include an analysis of domain names that have already been observed in order to identify newly registered domains. This task could be done quickly from a database that included all of the domain names observed on the SIE data feed, for example. The current text file based format has outgrown the size limits such a clumsy implementation should have. These newly observed domain names could then be checked against various sources for evil or benign sites.

One source of evil sites would be the fast-flux detection algorithm discussed in section 3.4. Spam filters, honey pots, and other malware collection or detection resources would also be cross-referenced. Other canonical block-list sites could also be checked, however these cannot be relied upon in the effort to fully automate the list and speed up the list if the goal is to improve upon the rate at which they report new sites. Sources for benign sites that the list would not want to block could continue to be used as the fast-flux detection algorithm already uses them.

Given the real-time nature of `Ncaptool`, it would be possible to update the list almost continuously for new content with only as much lag as is introduced by the SIE data feed, its processor, and external link speed. Given the implications of the phishing detection studies in section 3.2 and the high impact of phishing activity on businesses and consumers, it is highly desirable to improve the response time of block-lists. If the analysis of block-list response time is even nearly correct, over half of the time that a phishing site is active is granted to it because it has not been put on a block list yet. Very soon after these lists are published, the site becomes inactive. This seems to indicate that the block-list distribution and updating infrastructure is working reasonably well, even though it could always update more quickly. If the data is tentatively taken as is, 75% of a phishing site's lifetime it is not on a block-list. This indicates the need for faster block listing. Refinement and integration of several techniques, including the automated fast-flux detection, would aid a more rapid listing process. One approach that could be fruitful to explore would be keeping track of IP addresses that host blocked domain names. If it were noticed that a particular IP address space was being used more frequently than others to host malicious content, either knowingly or unknowingly, these IP spaces could be monitored more closely in the future, hopefully leading to improved detection rates of malicious URLs being hosted in the IP space. DNS is the natural protocol with which to track this activity.

4.2 FUTURE ANALYSIS OF PHISHING RESPONSE TIME

The future of the actual DNS based analysis of phishing block-list response time is highly uncertain. Since the last month's worth of data collection yielded no actionable data, it may seem that the undertaking is not worthwhile. However, the data does provide insight into the current state of phishing activity, and does so in near real time. The future usefulness of the technique will largely depend on the accuracy of predictions made in chapter 5 about what other sources will publish. If the data proves to be a useful indicator of trends, at the very least it would be wise to incorporate it into current phishing analysis frameworks. Nothing would be lost, and if it begins to provide actionable data on the efficacy of block-listing organizations, which would happen if the ratio of quickly found sites returns to near interval 1 levels, then new and unique information would be gained.

If these predictions provide several false positives, then a new approach to acquiring this data should be taken. In minimizing the lifetime of phishing sites it is important to understand which parts of the life cycle and defense cycle have the largest areas of improvement so as to focus industry efforts on these areas. This analysis has been one attempt to do just that, but if it proves not to be repeatable a different approach to the analysis ought to be undertaken. `Ncaptool` and its successors being designed at the ISC will remain valuable tools for this analysis, since they can provide the data in real time.

One further interesting project would be to compare the efficacy of various phishing block-listing organizations. If one could be shown to be significantly more effective at discovering phishing sites than others, there would be clear evidence to use that list over other available lists. Furthermore, these algorithms could be used to provide daily feedback to a list publisher as to its performance for the previous day. Since the requisite analysis requires about a day's computation from when the list is published, the publisher could get numbers for how well it did on a particular list very quickly, and if it notices that its performance is beginning to slip, could take actions to correct that reduction in performance before it becomes unmanageable.

5.0 CONCLUSION

DNS traffic analysis is an area of network situational analysis that is ripe for further development. The projects described in this thesis generally are describing proof of concept work, rather than a refined product. Section 3.1 provides evidence that DNS traffic is rich in information and that public data can be leveraged to build a comprehensive map of a domain with relatively little effort. The uses of this data are many and varied, and the familiar file-browser interface structure makes it easy for users and administrators to explore data and discover uses that will suit their needs.

Section 3.4 demonstrates the limits of DNS traffic analysis and where it would be better to consider analysis of more traditional targets such as bit rates or HTTP traffic. Unlike the analysis in the rest of chapter 3 in which DNS bears the burden of providing information alone, the information desired in section 3.4 cannot be acquired from analysis of DNS alone. Currently, tools such as the System for Internet Level Knowledge (SiLK) allow users to analyze and aggregate a large amount of network flow data, however they ignore DNS data (CERT 2009). Questions raised in section 3.4, as well as the fast-flux data from 3.3, would enhance the functionality of such comprehensive tools and DNS could provide much more sensible answers if it were juxtaposed with the rest of the flow data.

There are multiple conclusions to draw from section 3.2. In regards to the interval 1 data, it appears that the time it takes a phishing site to be listed on a block list is the bottleneck point in quickly taking down phishing sites. This conclusion is gleaned from a minority percentage of the total corpus of phishing sites, however even if it only describes malicious registrations and not phishing sites of all procurements it would be generally helpful to the Internet community if block-listing organizations could respond more quickly. Until future research manages to better identify the response time, these values are the only known measures of the response time of organizations to block-list phishing sites. Therefore, despite the uncertainty, the data provides a unique piece of information in the task of reducing malicious activity on the Internet.

There are several hypotheses as to why the nature of the data changed so much for the interval 2 data. One hypothesis is that when the APWG publishes is biannual report on phishing that will include the months of February and March, i.e. interval 2, that the report will find a much reduced presence of maliciously registered phishing sites and a much higher percentage of either hacked sites or sites registered with a static IP address. The data from the automated fast-flux detection could also be interpreted to support this hypothesis. The fast-flux detection algorithm primarily detects newly registered fast-flux domains, and it recorded a sharp drop in the number of such domains detected during interval 2.

There are a few easily conceivable reasons for this shift away from phishers registering domain names themselves. It is possible that registrars have made a concerted, successful effort to vet those who wish to register a domain name, making it economically inefficient for a phisher to attempt to register the domain. On the other hand, it is possible that a vulnerability in a common server distribution has gone unnoticed and has made it exceedingly easy for a hacker to gain access to a server and make it into a phish-hosting machine. This seems unlikely, because the high levels of abnormal activity associated with a massive vulnerability usually do not go unchecked for a month, however it is possible. Whatever the reason, if the shift is identified within the APWG report, then the interval 2 data has provided some insight into the nature of phishing. However this result is far short of what was hoped to be learned by the attempt at duplicating the interval 1 data.

Another hypothesis for the disparity between the distributions of the interval 1 and 2 data is that the block list organization's ability to find and list these domains decreased markedly between August and February. Some aspect of the community effort that was collecting these URLs could have shifted, and the organizations ability to report quickly on hacked domains fell off sharply. This would be a very dangerous turn of events, since potentially millions of dollars hang in the balance over even small changes in the efficacy of blocking phishing sites. If the appropriate APWG report does not discover a marked decrease in phishing registrations, then this becomes the leading hypothesis.

APPENDIX A: Glossary of Terms and Acronyms

APWG – Anti-Phishing Working Group. See <www.apwg.org>.

BPF – Berkley Packet Filter.

CERT – Computer Emergency Response Team. See <http://www.cert.org>.

DDoS – Distributed denial of service.

DITL – Day in the Life of the Internet. See <http://www.caida.org/projects/ditl/>.

DNS – Domain Name System. See <http://www.zoneedit.com/doc/rfc/> or <http://www.dns.net/dnsrd/rfc/> for the list of RFC's in which the Domain Name System is described.

HTTP – Hypertext Transfer Protocol

IP – Internet Protocol.

ISC – Internet Systems Consortium. See <www.isc.org>

Ncap – Primary program used by the SIE to share DNS data amongst consumers and to facilitate this research. It is available at <http://ftp.isc.org/isc/ncap/>

RFC – Request for Comments. Managed by the Internet Engineering Task Force (IETF). See <http://www.ietf.org/rfc.html>.

SEI – Software Engineering Institute. Operated by Carnegie Mellon University.

SIE – Security Information Exchange. Operated by the Internet Systems Consortium (ISC).

SiLK - System for Internet Level Knowledge. A network traffic flow analysis tool developed by the CERT-CC

TCP/IP – Transport Control Protocol / Internet Protocol. Two distinct protocols that are used extensively in delivering packets of information on networks.

TLD – Top level domains of the internet’s domain name structure, such as .com, .edu, and .uk.

UDP – User Datagram Protocol.

APPENDIX B: Bash Scripts

The following are bash scripts that were used to maintain and manipulate the ncap data. They are presented in no particular order, however are grouped into scripts with related functions.

All code in this appendix is subject to the following license. Please note that the owner of the code copyright is the Software Engineering Institute and Carnegie Mellon University, as the code was written largely under their employ.

INFORMATIONAL CONTENT, NONINFRINGEMENT, OR ERROR-FREE OPERATION. CARNEGIE MELLON UNIVERSITY SHALL NOT BE LIABLE FOR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, SUCH AS LOSS OF PROFITS OR INABILITY TO USE SAID INTELLECTUAL PROPERTY, UNDER THIS LICENSE, REGARDLESS OF WHETHER SUCH PARTY WAS AWARE OF THE POSSIBILITY OF SUCH DAMAGES. LICENSEE AGREES THAT IT WILL NOT MAKE ANY WARRANTY ON BEHALF OF CARNEGIE MELLON UNIVERSITY, EXPRESS OR IMPLIED, TO ANY PERSON CONCERNING THE APPLICATION OF OR THE RESULTS TO BE OBTAINED WITH THE DELIVERABLES UNDER THIS LICENSE.

Licensee hereby agrees to defend, indemnify, and hold harmless Carnegie Mellon University, its trustees, officers, employees, and agents from all claims or demands made against them (and any related losses, expenses, or attorney's fees) arising out of, or relating to Licensee's and/or its sub licensees' negligent use or willful misuse of or negligent conduct or willful misconduct regarding the Software, facilities, or other rights or assistance granted by Carnegie Mellon University under this License, including, but not limited to, any claims of product liability, personal injury, death, damage to property, or violation of any laws or regulations.

Carnegie Mellon University Software Engineering Institute authored documents are sponsored by the U.S. Department of Defense under Contract F19628-00-C-0003. Carnegie Mellon University retains copyrights in all material produced under this contract. The U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce these documents, or allow others to do so, for U.S. Government purposes only pursuant to the copyright license under the contract clause at 252.227.7013.

B.1 MAINTAINENCE OF CONTINUOUS NCAPTOOL CAPTURE

Command to run the continuous Ncaptool capture, which creates a new file every 2 minutes:

```
ncaptool -l 10.16.5.255/7433 -o /home/jspring/RollingCapture/tmp/RC -k 'gzip -9 ' -c 200000
```

This command must be run once to initiate the capture. After that, the other scripts take care of maintaining the files this command produces. It will run until it is forcibly killed from the command line. The following crontab commands organize this activity. Each script is included in a following subsection.

```
4 23 * * * /home/jspring/MaintainDirectoryCreation.sh #create directory for tmrw's captures and zip up unzipped files.
0 */6 * * * /home/jspring/MaintainDiskSpace.sh #check every 6 hours if disk usage is too high to continue
*/10 * * * * /home/jspring/MaintainRollingCapture.sh # ensure ncaptool is still running, and mv files from /tmp to today's dir.
```

B.1.1 MaintainDirectoryCreation.sh

```
CAPDIR=/var/rollingcapture
mkdir $CAPDIR/`date -d "tomorrow" +"%Y%m%d"`
# Create tomorrows directory.

gzip -9q $CAPDIR/tmp/*.ncap 2>/dev/null
```

```
# sometimes ncap files don't get zipped up properly, for whatever reason. gzip will surpress
          errors if no files exist to zip and any other errors to stderr are rerouted to null.
```

B.1.2 MaintainDiskSpace.sh

```
## This script is to be run a few times daily in conjunction with a continuous Ncaptool
          process that brings down packets.  It will delete the oldest directories so that new
          data can be captured without interference.  If directories are continuously being
          removed, look for another source of files that are overflowing the system's data
          stores

CAPDIR='/var/rollingcapture'
percentFull=`df -h | grep -o "[0-9]*% /$" | sed 's-% /--'`
  #gets the disk usage of the root dir (/ at end of line)

if [[ $percentFull -gt 91 && `ls -1 -D /var/rollingcapture | wc -l` -gt 2 ]]; then
# require at least 2 files to be in directory, so directory itself isn't removed
   toTrash=`tree -d $CAPDIR/ | head -2 | grep -o "2[0-9]*"`
   #since the directories are named by date in such a way as the oldest one is alphabetically
          sorted first, this will select the oldest directory.

   gunzip -c $CAPDIR/$toTrash/*.gz | /var/rollingcapture/ncapappend - |
          /usr/local/bin/ncaptool -n - -g - -m -e"
" "dns flags=aa type={a,aaaa}" | egrep -h ",IN,A{1,4}," | sed -r -e 's-^[0-9]* --' -e 's-([^0-
          9])0$-\1-' -e 's-,[0-9]{1,},?-,-g' -e 's-[[:space:]]--g' | tr "[[:upper:]]" "[[:lower:]]" | sort
          -u | gzip -9 > $CAPDIR/answerSummary.$toTrash.txt.gz
##This is one command.  Ncaptool does not recognize –e ”\n” so a literal new line must be
          used.  It copies all of the answer records from the day that is about to be deleted,
          cleans them up and standardizes the case, and stores the unique records for the day.
## Known error: This sed script does not always clean up every A record properly.
          Sometimes it does not remove all trailing 0’s, which are a remnant of Ncaptool
          formatting and not part of the A record.  It does remove them most of the time.

   rm -R $CAPDIR/$toTrash
   echo "$toTrash removed and unique answer RRs sent to a txt.gz file, collection will
          continue"
else
  echo "Sufficient space to continue collection, $percentFull % of space used."
fi
## These echos are sent to crontab, which can be configured to email them to the
          administrator.
```

B.1.3 MaintainRollingCapture.sh

```
if [ ! "$(/sbin/pidof ncaptool)" ] ; then
  nohup /usr/local/bin/ncaptool -l 10.16.5.255/7433 -o /var/rollingcapture/tmp/RC -k 'gzip -9
          ' -c 200000 &
  echo "ncaptool process restarted at `date`"
fi
# if ncaptool is not running (attempt to find process ID of it fails), restart it.

for file in $(ls /var/rollingcapture/tmp/*.ncap.gz); do
  TIMESTAMP=`echo $file | cut -d "." -f2`
  mv $file /var/rollingcapture/`date -d "1970-01-01 $TIMESTAMP sec" +"%Y%m%d"`/
done
# retrieve NCAP generated timestamp out of file name (in seconds since epoch), convert it
          to directory name (YYYYMMDD) and move file there.
```

B.2 DOMAIN MAPPING SCRIPTS

The third script here just calls the first two in the correct order and with some options for the scope of the search. These scripts utilize the fact that a file system has a structure similar to that of the domain name system and uses the native GUI for exploring file systems within an operating system to view the result of the sorting. If the sorting is done on a non-graphical interface operating system, the tarball that is created by the final script can be copied to a machine that does.

B.2.1 BestMapOfDomainPrep.sh

```
CAPDIR=/var/rollingcapture/

startTime=`date -d "2009-06-17" +%s`  #date Rolling capture was started, in seconds
endTime=`date  +%s`  #now, in seconds since 1970
#this captures all possible packets.  To narrow search, reassign from user or manual input
```

```
startTime=`date -d "2009-06-21 09:02:02" +%s`
#endTime=`date -d "2009-06-21 02:08:05" +%s`

JobStart=`date`
TLD=”.su$”

# if TLD regex should only include those records that have TLD as top domain, end string
        with '$', i.e. .co.uk$
if [[ $1 != "" ]]
  then TLD=$1
fi
if [[ $2 != "" ]]
  then startTime=`date -d "$2" +%s`
fi
if [[ $3 != "" ]]
  then endTime=`date -d "$3" +%s`
fi
#override defaults with user input, if it is supplied

find $CAPDIR -type f -print  | awk -v ST="$startTime" -v ET="$endTime" ' BEGIN
        {FS="."} {if ($2 >= ST && $2 <= ET) print $0} ' > searchSpace.$TLD.txt
#the second 'column' of the file name is the time in s since the epoch, per standard NCAP
        formatting
count=0
echo "`wc -l searchSpace.$TLD.txt` files will be searched through"

query=1
qbit=$4
if [[ $4 != "" && ${qbit:0:1} == 'q' || ${qbit:0:1} == 'Q' ]]
    then query=0 # if user inputs anything that begins w/ q or Q to query-option field, then set
        qname search flag to 0 (true)
fi
# if the query flag is set to 0, search only the qname.  Otherwise, use a regex to search the
        files.
if [[ $query -ne 0 ]]
  then
  TLDregex=`echo $TLD | tr '.' '\.'` # Any .'s in Domain name should be treated as literal
        dots in the search, '\' will escape them in regex.
  cat searchSpace.$TLD.txt | while read LINE
  do
    ##unzip the file first (temporarily, cat will allow such that the file isn't modified), each
        file is on one line in the search Space
    cat $LINE | gunzip -c | ncaptool -n - -g - -m -e"
" "dns regex=$TLDregex" >> tmpBestMapOfDomain.$TLD.ncap.txt
```

```
  # read unzipped file from standard in and output txt to standard out, or file
    let "count += 1"
    echo -n $count, #simply so user knows how many files have been searched
  done
else
  cat searchSpace.$TLD.txt | while read LINE
  do
    ##unzip the file first (temporarily, cat will allow such that the file isn't modified), each file is on one line in the search Space
    cat $LINE | gunzip -c | ncaptool -n - -g - -m -e"
" "dns qname=$TLD" >> tmpBestMapOfDomain.$TLD.ncap.txt
    # read unzipped file from standard in and output txt to standard out, or file
    let "count += 1"
    echo -n $count, #simply so user knows how many files have been searched
  done
fi


## search/manipulate the output, conglomerated text/binary file here
grep -h ",\(\(IN\)\|\(TYPE41\)\)," tmpBestMapOfDomain.$TLD.ncap.txt | sed -r -e 's-^[0-9]*--' -e 's-([^0-9])0$-\1-' -e 's-,[0-9]{1,},?-,-g' -e 's-[[:space:]]--g' | tr "[[:upper:]]" "[[:lower:]]" | sort -u > Prepared.$TLD.txt
# Type 41 is necessary explicitly b/c OPT records use the class field in a novel way and ncaptool doesn't recognize them as IN class
# normalizes the Ncaptool output into a more usable format, removing some formatting bits that can't be opted out of from the start.
# Eliminates duplicate records, regardless of some misceleneous timestamps (sed to remove [0-9] between ,'s)
# only operating on the entries in the RR's of the packets (initial grep), which is the prefered unit of study for this approach.

rm searchSpace.$TLD.txt

echo "the job started at $JobStart and now it ends at `date`"
```

B.2.2 DomainGUIprep.sh

```
CAPDIR=/var/rollingcapture/ # not used here, but should match where BestMapOfDomainPrep.sh is looking
PREPFILE=prepared.txt # the input file. This default must be overridden, it is obsolete.

if [ -e $1 ] # user input is a file that exists
```

```
  then PREPFILE=$1
  TLD=`echo $1| sed -r -e 's-Prepared\.{1,2}--' -e 's-\.txt--' -e 's-\\$--g' `
  echo $TLD # testing code
  # remove formatting from BestMapOfDomainPrep.sh
else
  echo "Please include a valid file of Prepared Resource Records for inspection"
exit 3 # The status it will return upon exiting is 3, which will be "invalid input file"
fi

STARTDIR=/home/jspring/analysis/$TLD
WorkDir=$STARTDIR
# working directory will server as something of a pointer to which DN the program is
            working with as well

#Defines three functions, one to determine which of the other 2 to call
# and one each to gather the information important for a host or zone, respectively
# the program is recursive, and is kicked off below these function definitions

function isDomain {
# We will define hosts by anything that has an IP address or CNAME associated with it.
# Everything else will be treated as a domain
code=2
if grep -q "^$1,in,\(\(cname\)\|\(aaaa\)\|a\)," $PREPFILE
  then code=0
fi
if grep -q ",ns,$10\?" $PREPFILE
  then code=0
fi
if grep -q "^$1,in,\(\(ns\)\|\(soa\)\|\(mx\)\)," $PREPFILE
  then code=1
fi
# will return 0 if it found something matching patterns that are indicative that the name is a
            host.
return $code
}

function buildPath {
  echo $1 |tr '.' "\n" | tac | while read LINE
    do name=$LINE.$name
    echo $name
    done | sed 's-.$-/-' | tr -d "\n" | sed 's-/$--'
  # for 'ns1.ischool.pitt.edu' outputs 'edu/pitt.edu/ischool.pitt.edu/ns1.ischool.pitt.edu/'
}

function forHost {
correctPath=$STARTDIR/`buildPath $1 | sed 's-/[^/]*$--'`
hostFile="$correctPath/~~HOST.$1.txt"
if [ ! -d $correctPath ]
  then mkdir -p $correctPath
fi
touch $hostFile
grep "^$1," $PREPFILE > $hostFile
grep ",$10\?$" $PREPFILE >> $hostFile
# intresting stuff will be where the name is at the beginning (cname, A, AAAA) or end (ptr, ns) of the RR line. optional '0' is for damn ncaptool formatting workaround
}

function forDomain {
WorkDir=$STARTDIR/`buildPath $1`
if [ ! -d $WorkDir ]
  then mkdir -p $WorkDir
fi
infoFile="$WorkDir/~~INFO.$1.txt"
touch $infoFile
grep "^$1," $PREPFILE > $infoFile
grep ",$10\?$" $PREPFILE >> $infoFile
# intresting stuff will be where the name is at the beginning (cname, A, AAAA) or end (ptr, ns) of the RR line. optional '0' is for damn ncaptool formatting workaround
# While loop will get all the recorded names exactly one sub-domain below the current domain. Assumes ASCII
# for unicode, perhaps use regex for '^(anything not a .)\.$1,' . untested, may be too general
grep -o "^[0-9a-z-]*\.$1," $PREPFILE | sed 's-,$--' | sort -u | while read LINE
do
  if [ -z "$LINE" ] # if the line is empty, then continue to next line
    then continue
  fi
     # echo $LINE #for testing
  isDomain $LINE # this if-else contains the potential for recursion
  if [[ $? == 0 ]]
    then forHost $LINE
  else
    forDomain $LINE
  fi
done

}

isDomain $TLD
if [[ $? == 0 ]]
  then forHost $TLD
else
  forDomain $TLD
```

```
fi
```

B.2.3 DomainMapStart2Finish.sh

```
CAPDIR=/home/jspring/RollingCapture/
EXCTDIR=/home/jspring/analysis

startTime=`date -d "2009-05-28" +"%Y-%m-%d %H:%M:%S"`  #date Rolling capture was
        started
endTime=`date  +"%Y-%m-%d %H:%M:%S"`  #now, as the default
#this captures the most packets.  To narrow search, reassign from user or manual input

TLD=.gov$

# if TLD regex should only include those records that have TLD as top domain, end string
        with '$', i.e. .gov$
if [[ $1 != "" ]]
  then TLD=$1
fi
if [[ $2 != "" ]]
  then startTime=$2
fi
if [[ $3 != "" ]]
  then endTime=$3
fi
#override defaults with user input, if it is supplied
# the script will handle converting the user inputs to its usable dates.

PrepStart=`date +"%Y-%m-%d %H:%M:%S" `
nohup $EXCTDIR/BestMapOfDomainPrep.sh "$TLD" "$startTime" "$endTime"

nohup $EXCTDIR/BestMapOfDomainPrep.sh "$TLD" "$PrepStart"
 # assumes that the original end time was 'now' and this will run again to check the packets
        collected while the first process was running.

nohup $EXCTDIR/DomainGUIprep.sh "Prepared.$TLD.txt"
 # create the map of the domain from the file created by BestMapOfDomainPrep.sh

TLDName=`echo $TLD | sed -e 's-^\.--' -e 's-\\$--g' `
tar -cf $EXCTDIR/$TLDName/$TLDName.tar $EXCTDIR/$TLDName/*
 # create a tarball of the directory tree just created to represent a map of the domain.
```

B.3 PHISHING DETECTION SCRIPTS

Several of the scripts used in the phishing analysis are idiosyncratic to the research and the sources of information that were used, and therefore not of general interest. The scripts included here should be of some use. For example, script B.3.1 takes as input an arbitrary list of domain names and dates that they were added to the list and will search through an arbitrary number of Ncap files, as organized per the scripts in B.1, and create an Ncap file with only queries about names in the list. This uses the qnames.so plugin. This can be used for further analysis about the names in the list, such as with script B.3.2, which will calculate the number of seconds between the first instance of the domain name in the DNS data and the date it was added to the list.

B.3.1 ManySiteSearch.sh

```
CAPDIR=/var/rollingcapture/

startTime=`date -d "2009-05-29" +%s`  #date Rolling capture was started, in seconds
endTime=`date  +%s`  #now, in seconds since 1970
#this captures all possible packets.  To narrow search, reassign from user or manual input

phishList=sorted.APWG.txt
# phishing list needs to be just a list of domain names as they would be queried for and
        times added to list

if [ -e $1 ] # if the user input filename exists.
  then phishList=$1
fi
if [[ $2 != "" ]]
  then startTime=`date -d "$2" +%s`
fi
if [[ $3 != "" ]]
  then endTime=`date -d "$3" +%s`
fi
```

```
#override defaults with user input, if it is supplied

find $CAPDIR -type f -print  | awk -v ST="$startTime" -v ET="$endTime" ' BEGIN
        {FS="."} {if ($2 >= ST && $2 <= ET) print $0} ' > searchSpace.$phishList.txt
#the second 'column' of the file name is the time in s since the epoch, per standard NCAP
        formatting
count=0
echo "`wc -l searchSpace.$phishList.txt` files will be searched through from $startTime to
        $endTime"

cut -f1 -d',' $phishList > tempphishList.txt

cat searchSpace.$phishList.txt | while read LINE
  do
    ##unzip the file first (temporarily, cat will allow such that the file isn't modified), each
        file is on one line in the search Space
    gunzip -c $LINE | ncaptool -n - -o - -D /usr/local/lib/qnames.so,-qftempphishList.txt |
        ncapappend all.$phishList.ncap
    # read unzipped file from standard in and output binary to Ron's append to file
    # echo $?
    let "count += 1"
    echo -n $count, #simply so user knows how many files have been searched
done

rm searchSpace.$phishList.txt
rm tempphishList.txt
```

B.3.2 checkEarliestDates.sh

```
phishList=sorted.APWGphish20090714.txt

if [ -e $1 ]
  then phishList=$1
fi
# input list must be of the format PhishingDomain,Date first seen(YYYY-MM-DD
        HH:MM:SS)
ncapFile=all.$phishList.ncap
# as long as naming convention in ManyPhishSiteSearch.sh is maintained this will work
if [[ -e $2 && $2 != "" ]]
  then ncapFile=$2
fi
 # allow for manual override of location of NCAP file.
touch $ncapFile
```

```
while read LINE
do
  Dname=`echo $LINE | cut -f1 -d',' `
  listTime=`echo $LINE | cut -f2 -d',' `
  firstTime=$(ncaptool -n $ncapFile -g - -e "_" "dns qname=$Dname" | awk ' BEGIN
            {earliestDay ="2999-12-31"; earliestTime = "23:59:59.999999999"}
    { thisDay = $4; thisTime = $5;
     if ( thisDay < earliestDay || thisDay == earliestDay && thisTime < earliestTime )
     { earliestDay = thisDay ; earliestTime = thisTime } }
    END { printf "%s %s", earliestDay, earliestTime } ' )

  if [[ $firstTime != "2999-12-31 23:59:59.999999999" ]]
    then echo -n "$LINE,$firstTime" >> checked.$phishList
    firstTime=`date -d"$firstTime" +%s `
    listTime=` date -d"$listTime"  +%s `
    let "difference=$listTime-$firstTime"
    if [[ $difference > 0 ]]
      then echo ",`date -d "0000-01-01 $difference sec" +"%Y-%m-%d %H:%M:%S" `" >>
            checked.$phishList
    else
      echo ",^No requests before phishing Listing" >> checked.$phishList
    fi
  else
    echo "$LINE,#No DNS packets found for this name" >> checked.$phishList
  fi

  echo `date -d "0000-01-01 $difference sec" +"%Y-%m-%d %H:%M:%S" `
done < $phishList
#read in the list of phishing sites, and for each find the packet which first asks for the
            domain by checking if the time
# on the packet is earlier than the earliest seen time.  Output this data to a new file to further
            analyze the data.
```

B.3.3 evaluateDNhack.sh

```
dataFile=checked.sorted.APWGphish20090727.txt
if [[ $1 != "" && -e $1 ]]
  then dataFile=$1
fi
 # the default should be overridden by user input, as long as it exists

# loop through the time segments that are most important for inspection
# the first day is broken up roughly into 5ths, and the rest are binned by days
# for each day grep to aquire the list of hosts found that many days distance from the zero
        hour
# and then for each of those hosts use wget to determine if it matches the hacked or
        registered pattern, and collect statistics, which go to stdout
for counter in '01 0[0-4]' '01 0[5-9]' '01 1[0-4]' '01 1[5-9]' '01 2[0-4]' 02 03 04 05 06 07 08
        09 `seq 10 22`
  do
  grep "0000-01-$counter" $dataFile | cut -f1 -d',' > DNs
  total=$((0))
  registered=$((0))
    # these will be the variables to keep track of the total sites for the day and number that
        appear to have been registered maliciously

  while read LINE
        do
        total=$((total+1))
        wget --tries=3 --spider --timeout=9 --dns-timeout=9 --no-dns-cache --no-cache --
        user-agent='Mozilla/5.0' $LINE 2>/dev/stdout | grep -B 1 ' 200 ' > tmpResponse

        if  [[ $? != 0 ]]
          then registered=$((registered+1))
        else
          grep -q "$LINE" tmpResponse
          if [[ $? != 0 ]]
        # If the response code was 200, but the name who made the response was not the
        name we asked about
        # i.e. there was a redirect or 404 page given, then the real page we were looking for
        was not found
        # therefore count it as though we didn't get a 200.
                then registered=$((registered+1))
          fi
        fi

        done < DNs
  echo "Considering differences of 0000-01-$counter days ,$registered/$total, sites appear to
        be registered maliciously"

done
```

BIBLIOGRAPHY

Aaron, G. a. R. R. (2009). Global Phishing Survey: Trends and Domain Name Use 1H2009 Lexington, MA, USA, Anti-Phishing Working Group.

Aaron, G. a. R. R. (2009). Global Phishing Survey: Trends and Domain Name Use 2H2008. Lexington, MA, USA, Anti-Phishing Working Group**:** 26.

Baker, W. H., C. David Hylender and J. Andrew Valentine (2008). 2008 Data Breach Investigations Report: A study conducted by the Verizon Business RISK Team, Verizon Business**:** 29.

CERT (2009). "Monitoring for Large-Scale Networks." Retrieved April 20, 2010, from http://tools.netsa.cert.org/silk/.

Cyveillance (2008). The Cost of Phishing: Understanding the True Cost Dynamics Behind Phishing Attacks. Arlington, VA, Cyveillance, Inc.

Faber, S. (2009). Uses of DNS analysis. J. Spring.

ISC (2009). "Security Information Exchange Channel Guide." Retrieved Feb. 10, 2010, from http://sie-gs2.sql1.isc.org/channelguide/index.php.

Kash, H. a. D. W. (2008). H-root DNS traces DITL 2008 (collection).

Kriegisch, A. (2009). Detecting Conficker in your Network. Vienna, Austria, National Computer Emergency Response Team of Austria.

McTurk, K. (2010). RE: .edu Comments/Feedback Form. J. Spring, Educause, Inc.**:** 1.

Mockapetris, P. (1983). RFC 883. Fremont, CA, Internet Engineering Task Force.

Moore, T. a. R. C. (2007). Examining the Impact of Website Take-down on Phishing. APWG eCrime Researchers Summit. Pittsburgh, PA, USA, APWG.

Perdisci, R., Igino Corona, David Dagon, and Wenke Lee (2009). "Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces." Proceedings of The 25th Annual Computer Security Applications Conference.

Riden, J. (2008). Know Your Enemy: Fast-Flux Service Networks, The Honeynet Project.

Vixie, P. (2009). "What DNS is Not." ACM Queue **7**(10): 6.

Weimer, F. (2005). Passive DNS Replication.

Wessels, D. a. M. F. (2003). Wow, That's a Lot of Packets. Boulder, CO; San Diego, CA, The Measurement Factory & CAIDA, San Diego Super Computing Center, University of California San Diego.

www.root-servers.org. "Root Server Technical Operations Association." from http://www.root-servers.org/.

www.spamhaus.org. "What is "fast flux" hosting?". Retrieved Feb. 9, 2010, from http://www.spamhaus.org/faq/answers.lasso?section=ISP%20Spam%20Issues#164.