UNDERSTANDING DNS-BASED CRIMINAL INFRASTRUCTURE FOR INFORMING TAKEDOWNS

A Thesis
Presented to
The Academic Faculty

by

Yacin Nadji

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
December 2015

UNDERSTANDING DNS-BASED CRIMINAL INFRASTRUCTURE FOR INFORMING TAKEDOWNS

rest in peace ol' dirty

ACKNOWLEDGEMENTS

Throughout my time at Georgia Tech, I've been fortunate enough to work with some of the smartest people I have ever met and learned a lot from them about computer security as well as life in general.

My advisors, Wenke Lee and Manos Antonakakis, were instrumental to my success. Wenke's wisdom, both in academic and personal matters, guided me through the Ph.D. process. Manos's help in designing projects, navigating the field, and the occasional lambastes when I wasn't working up to my potential not only made me a strong researcher, but helped me become a man.

I was fortunate enough to work with Roberto Perdisci and David Dagon during my time here. If something passed under Roberto's insightful eyes and was deemed decent, it was likely to be accepted by the community at large. David's brilliant brainstorming and word smith skills got me excited to be working in security and let me explain our work to the community at large.

My research has always been data driven, and without the data and people at Damballa, Inc. I would not have come out the other end of the program. Gunter Ollmann taking me on as an intern helped set me along a successful path. Marshall Vandegrift, Kevin Smith, and Jeremy Demar made my work possible by answering my dumb questions and providing their knowledge on criminal threats and infrastructure.

The support of GTISC and Astrolavos labs' faculty and students played no small role in my tenure here. Thanks to Charles Lever and Yizheng Chen for being wonderful teammates and Kapil Singh, Brendan Dolan-Gavitt, Paul Royal, and Martim Carbone for helping me feel at home when I began my journey here. My committee members—Michael Bailey, Mustaque Ahamad, and Doug Blough—gave useful and insightful guidance on my proposal and dissertation.

The Information Retrieval Lab at Illinois Tech (now at Georgetown) sparked my interest in research and taught me a lot about getting things done. Special thanks to Ophir Frieder for advising me throughout undergrad, and Jordan Wilberding for taking a chance on a punk like me.

My emotional support structure helped me survive the grueling nights. I am indebted to you, Mom and Baba, for dealing with my temper tantrums as a child and molding me into the halfway decent person I am today; sahit. I'd like to give a shout out to my crow squad: Robert Pienta, Taoran Le, and Casey Battaglino for keeping me sane; cawww!

Last, but not least, big ups to OutKast, the Wu-Tang Clan, and 2Pac for giving me southern pride, great names for my systems, the inspiration to achieve great things, and on point music to bump to while rolling around Atlanta. Wu-Tang is for the children; hootie hoo.

My sincerest thanks to all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Botnets are a pervasive threat to the Internet and its inhabitants. A botnet is a collection of infected machines that receive commands from the botmaster, a person, group or nation- state, to perform malicious actions. Instead of "cleaning" individual infections, one can sever the method of communication between a botmaster and her zombies by attempting a botnet takedown, which contains the botnet and its malicious actions.

Unfortunately, takedowns are currently performed without technical rigor nor are there automated and independent means to measure success or assist in performing them. This dissertation focuses on understanding the criminal infrastructure that enables communication between a botmaster and her zombies in order to measure attempts at, and to perform, successful takedowns. We show that by interrogating malware and performing large-scale analysis of passively collected network data, we can measure if a past botnet takedown was successful and use the same techniques to perform more comprehensive takedowns in the future.

CHAPTER I

INTRODUCTION

Botnets are a pervasive threat to the Internet and its inhabitants. A *botnet* is a collection of infected machines that receive commands from its *botmaster*—a person, group or nation-state—to perform malicious actions. These malicious actions can range from attacking individuals—stealing usernames, passwords, and credit card numbers—to those with international, geopolitical consequences, like sabotaging nuclear refineries [55]. To better understand these threats, researchers have examined the *command and control* (C&C) servers used by botmasters to control their infected machines, or "zombies."

C&C servers are hosted on *criminal* or *malicious infrastructure* that enable the botmaster to reach her victims. Malicious infrastructure is comprised of *network assets*, such as domain names and IP addresses, that the botmaster uses to relay messages. Infrastructures can be simple—a single, static IP address—contain a layer of agility—a set of domains that resolve to a changing set of IPs—or be more agile still and rely on algorithmically generated domain names or a peer-to-peer overlay network.

In addition to remediating individual infections the security community has recently turned to performing *botnet takedowns*, or disabling the C&C servers used by the attackers to issue commands to their zombies. Remediation of infected end-users is a difficult and time-consuming process, whereas a botnet takedown has some of the same benefits and can create a greater impact in a shorter period of time. Specifically, zombies that cannot reach their botmaster will not participate in further malicious activities. Unfortunately, takedowns have up to this point largely been driven by manual processes.

Although some earlier takedown attempts have been successful [38, 91], many have been incomplete due to false negatives during infrastructure enumeration [113, 112], or caused significant collateral damage due to lack of specificity [112, 62]. This shows the community needs *automated techniques* to enumerate malicious infrastructure with specificity to reduce false negatives and false positives for future takedowns of DNS-based botnet infrastructure. Furthermore, automation will make it possible to evaluate previous takedowns' success, further reduce false positives by identifying collateral damage, and allow the community to keep pace with agile attackers in the event a takedown does not succeed.

Historically, takedowns have failed because of false negatives and/or false positives. In the false negative case, the performers of a takedown fail to comprehensively identify and take down all the criminal infrastructure used to support the botnet's operations. For example, in Microsoft's takedown of a Zeus botnet [113] many C&C domains were not deactivated and the zombies could continue their malicious activity. In the false positive case, benign infrastructure is taken down in addition to the malicious infrastructure, often due to the takedown lacking specificity. For example, the Bladabindi takedown [62] seized control of 23 domain names, which were the parent zone for over 5 million fully qualified domain names. Some of these 5 million were known to be used by Bladabindi, but the takedown affected all domains under the 23 zones. Had the takedown targeted the malicious domains specifically, the 18 million customers would have likely been less damaged. In some cases, a takedown fails due to both false negatives and positives, as we show in the 3322.org/Nitol takedown in Section 5.2.4.

To automatically enumerate infrastructure in preparation for a takedown three tasks must be performed: malware-based infrastructure enumeration, network-based infrastructure enumeration, and to identify potential collateral damage. Using the output from these tasks, we can evaluate the success of a historic takedown and recommend actions for a future takedown. Each of these components comprise the contributions of this dissertation.

1.1 Contributions

Malware Infrastructure Enumeration To perform a proper takedown we must interrogate the malware to identify any *alternative plans* they may contain after their primary infrastructure is disabled. This could be as simple as additional network assets hardcoded into the malware, to as complex as a completely different infrastructure and protocol, for example a domain-name generation algorithm (DGA) or peer-to-peer (P2P) C&C protocol. Disabling the malware's primary infrastructure is clearly insufficient if the malware author has programmed in backup behavior that has not been taken into account.

We built `gza`, which interrogates malware to reveal alternative execution plans of malware and classify instances where the backup behavior is more elaborate, e.g., DGA- or P2P-based C&C scheme. `gza` operates by rewriting network packets in an RFC compliant way to "play games" with malware and trick it into believing its primary infrastructure has been disabled. Without coordination, which itself would use the network and could thus be gamed, the games are indistinguishable from real takedown actions from the perspective of the executing malware. Furthermore, this approach is agnostic to the underlying malware analysis platform and easy to extend to handle other C&C protocols.

`gza` can coerce hundreds of thousands of likely malicious domain names from malware that never appeared on public blacklists after our study and that sophisticated backup C&C techniques, such as DGA or P2P, can be reliably detected by playing games with malware. Both are necessary to perform a successful takedown. `gza` is available as open-source software [1] as a module to the popular malware analysis framework "Cuckoo Sandbox."

Network Infrastructure Enumeration If only a portion of a botnet's malicious infrastructure is taken down, the takedown will not succeed. In fact, some argue that performing partial takedowns makes the Internet *less safe* as it trains botnet operators to design more resilient criminal infrastructure []. The goal is to increase the understanding of malicious infrastructure by analyzing real-world network data.

We built `deck` to enumerate malicious infrastructure based on an initial set of *seed domains*, i.e., initial knowledge of a set of infrastructure, by analyzing the *passive DNS* history of the seed domains. `deck` builds and visualizes graph representations of malicious infrastructure and analyzes them with respect to performing comprehensive takedowns using network analysis measures. We show previous work that focuses on "bad" autonomous systems is unlikely to help combat Internet miscreants as their malicious infrastructure is typically distributed and that, depending on the network structure, different takedown approaches can cause more serious harm to the botnet. We found that in many cases criminal networks can be disabled by de-registering as few as five domain names. In more sophisticated networks, targeting the C&C servers themselves can cause substantial damage to the infrastructure. In one case, disabling 20% of a criminal network's hosts would reduce the overall volume of successful connections by 70%, suggesting that certain "critical" nodes are more important to the infrastructure.

`deck`'s visualization of criminal networks has been implemented into the product offering of Damballa, Inc.

[1] `https://github.com/ynadji/cuckoo/tree/gza`

Measuring & Recommending Takedowns Currently, takedowns are performed in an ad-hoc manner without oversight nor a method for evaluating success after the fact. This encourages perfunctory takedowns to drum up positive press and inhibits the evaluation of techniques that do and do not work, which prevents the security community from advancing in our fight against criminal activity on the Internet. Without a reliable process to measure and perform takedowns the community is doomed to make the same mistakes over and over again.

We built `rza`—a combination of `gza` and `deck`—to measure the success of historic takedowns and use a similar methodology to recommend how to perform a takedown and what precautionary measures must be taken to ensure the takedown is successful. We find that several historic takedowns were performed without fully enumerating the criminal infrastructure, allowing malicious activity to continue unfettered. We also show many threats active at the time of the study can be disabled easily by only "sinkholing" or disabling a handful of domain names. Finally, many of the studied takedowns caused serious *collateral damage* to legitimate customers; this motivates the need to build tools to identify network assets likely shared between benign users and Internet miscreants to ensure a takedown action does not cause undue harm.

The output of `rza` for several botnets has been given to the FBI to inform on the backup behavior of malware and the feasibility of taking down their criminal infrastructure.

Collateral Damage—Advanced Threats While studying takedowns using `rza`, we noticed a common instance of collateral damage occurs when attempting to take-down advanced persistent threat, or APT, infrastructure. APT infrastructure commonly makes use of compromised machines for its criminal infrastructure as it provides plausible deniability for attacks with potential geopolitical consequences, among other things. Taking down such dangerous criminal infrastructure is important to safeguard enterprises and governments around the world, but cannot be done at the cost of harming legitimate users.

We present `ghost&rae`, a system to model "manual APT" infrastructure that commonly resides on compromised hosts alongside benign domains. After classifying these domains, we cluster them to separate malicious APT domains from benign domains to prevent them from being taken down and disabled as collateral damage. We show that `ghost&rae` can model APT domains with high true positive and low false positive rates. While this is only one class of collateral damage to detect, it represents an interesting case as it is a common source of false positives for existing reputation systems [4, 14].

`ghost&rae` is currently being tested for production deployment at Damballa, Inc.

1.2 Dissertation Overview

The remainder of this dissertation is structured as follows. Chapter 2 contains the necessary prior work and background to understand the dissertation's content, as well as the datasets and notation used throughout the remaining chapters.

Chapter 3 describes `gza`, the system to perform malware-level infrastructure enumeration. Section 3.1 describes how `gza` "plays games" with malware during live execution to trick the sample into believing its primary infrastructure is disabled. Section 3.2 describes the architecture and implementation of `gza` in detail. Section 3.3 describes the experiments used to evaluate `gza` and how it can be used to identify higher level backup behaviors that must be known in order to perform an effective takedown, while Section 3.4 describes the results of these experiments.

Chapter 4 describes `deck`, the system to perform network-level infrastructure enumeration. Section 4.1 describes how criminal network graphs are constructed to represent the enumerated infrastructure. Section 4.2 describes the properties of these graphs in general while Section 4.3 presents four "interesting" criminal networks and their properties. Both sections focus on the feasibility of takedown and in the case studies, simulate the damage done to the infrastructure by revoking domain names or IP addresses.

Chapter 5 presents `rza`, which combines `gza` and `deck` in order to evaluate past takedowns and recommend if and how a future takedown should be performed. Section 5.1 presents how the two aforementioned systems are used in concert to study and recommend takedowns. Section 5.2 analyzes three historic takedowns in depth to determine if they were successful or not. Section 5.3 shows how `rza` can be used to identify which botnet infrastructure can be safely disabled through sinkholing or domain de-registration and which employ more sophisticated forms of command and control.

In Chapter 6 we introduce `ghost&rae`, a system to model and cluster APT infrastructure and separate malicious domains from the benign domains they share hosting with. Section 6.1 explains background specific to APT attacks, as well as defining what is considered an APT in this dissertation. Section 6.2 provides an overview of `ghost&rae` with details of the supervised and unsupervised components of the system explained in Section 6.3. `ghost&rae` is evaluated using standard machine learning techniques in Section 6.4 and interesting case studies of APTs discovered in the wild are presented in Section 6.5.

Chapter 7 discusses the potential for evading the proposed system given that the techniques are known to the public and how quickly these techniques can respond to an attacker with agile infrastructure. Section 7.1 describes potential evasions for each of the presented systems and the likely costs attackers face when attempting to evade them. Section 7.2 explains how we can handle an agile attacker. Even if a takedown brought on by our systems is incomplete, the time to re-run the systems is on the order of hours; quick enough to place even the fastest attackers back under scrutiny and the pressure of a looming takedown. Finally, we conclude in Chapter 8.

CHAPTER II

BACKGROUND & PRIOR WORK

In this chapter we describe the background necessary to understand the this dissertation and relevant related work. First, we provide background on the current process for performing botnet takedowns and describe past successes and failures. Next, we explain the datasets that are used throughout the dissertation to address the current limitations in performing and measuring botnet takedowns. We conclude this chapter with details of the relevant related work.

2.1 Background

2.1.1 Botnets and Takedowns

Botnet takedowns are not uncommon, and may take many different forms. Considering the heterogeneous nature of client machines and the difficulty in keeping individual machines clean from infection, taking down the botnet C&C is an attractive alternative. A successful takedown eliminates most external negative impacts of the botnet, effectively foiling further attacks (e.g., spam, DDoS, etc.) by the infected hosts, which can number in the millions. In the past, takedowns have been performed by revoking sets of C&C IP addresses from hosting providers, de-peering entire Autonomous Systems (AS), or, more recently, sinkholing or revoking C&C domains.

Conficker is an Internet worm that infected millions of computers and remains one of the most nefarious threats seen on the Internet to date [38]. Conficker's latter variants employed a DGA that would generate 50,000 pseudo-random domain names every day to communicate with its C&C server. The takedown of Confiker required immense coordination across hundreds of countries and top-level domains (TLDs), and numerous domain registrars and registries. The takedown efforts were coordinated by the Conficker Working Group (CWG) [38]. The takedown required reverse-engineering the malware binaries, and reconstructing the DGA. Then, the CWG pre-registered all 50,000 domains per day that could potentially be used for C&C purposes, thus preventing the botmaster from regaining control of the bots. The success of CWG's efforts highlight the importance of participation and support from key governing and regulatory bodies, such as ICANN, and the need of cooperation between the private sector and governments around the world.

Mariposa, a 600,000-strong botnet of Spanish origin, provides another example of a takedown operation initiated by a working group that relied on sinkholing known malicious domains. Interestingly, Mariposa's botmasters were able to evade a full takedown by bribing a registrar to return domain control to the malicious operators [91], underscoring the fact that barriers to successful takedowns are not only technical ones.

The DNSChanger [177] "click-jacking" botnet was also taken down through a working group. DNSChanger altered upwards of 300,000 clients' DNS configurations to point to rogue DNS resolvers under the control of the attackers. This allowed the attackers to direct infected hosts to illegitimate websites, often replacing advertisements with their own to generate revenue. DNSChanger had to be taken down by physically seizing the botnet's rogue DNS servers. The takedown was accomplished in late 2011. Largely considered successful, the DNSChanger once again shows the importance of collaboration when performing comprehensive takedowns.

Not all takedowns are performed at the DNS-level, however, as shown in the takedowns of McColo [89], AS Troyak [109], and other "bulletproof hosting providers," or networks known to willingly support malicious activities. These are extreme cases where the networks in question essentially hosted only malicious content, and removing the entire network would disable large swaths of botnets and related malicious network infrastructure. The effect of these takedowns were indirectly measured by witnessing drops in spam levels, for example, upwards of two-thirds decrease after McColo's shutdown [90]. Unfortunately, if a particular botnet relied on the DNS to perform C&C resolutions into these bulletproof networks, once a new host was provisioned the threat would continue. Sure enough, we saw spam levels rise back to normal levels as botnets moved to other hosting providers [69].

2.1.2 Datasets

This dissertation relies on large datasets to understand botnet infrastructure and the malware the botnet's use, and identify likely sources of collateral damage. On the network-side, a large passive DNS dataset is used to identify historic relationships between botnet infrastructure and to create features to model APT infrastructure. On the system-side, we have a large corpus of malware samples, as well as a database of domain names and IP addresses a given malware sample communicated with during its dynamic execution in a malware analysis environment. This is used to identify malware samples to interrogate, and identify domains that are likely to be used for malicious purposes.

Passive DNS A passive DNS (pDNS) database stores historic mappings between domain names and IP addresses based on successful resolutions seen on a live network over time. pDNS databases allow us to reconstruct the historical structure of DNS-based infrastructure based on how it was used by clients. Our pDNS is constructed from real-world DNS resolutions seen in a large North American ISP, as well as multiple enterprise companies. This allows us to identify the *related historic domain names* (RHDN) for a given IP, namely all domains that resolved to that IP in the past. Also, pDNS allows us to find the *related historic IP addresses* (RHIP) for a given domain name, i.e., all the IPs to which the domain resolved to in the past. Furthermore, the RHIP/RHDNs can be limited to domain-to-IP mappings that occurred during a particular time frame of interest, thus allowing us to focus on the crucial days before and after a takedown took place.

To enable our takedown analysis we define the following functions over the pDNS database:

- RHIP(`domain`, `start_date`, `end_date`): returns all domains historically related to the `domain` argument over the period between the desired start and end dates. For example, RHIP(`foo.com`, 2012/01/01, 2012/01/05) would return the set of all IP addresses `foo.com` successfully resolved to between January 1^{st}, 2012 and January 5^{th}, 2012, inclusive.

- RHDN(`IP`, `start_date`, `end_date`): similarly, RHDN returns all domains historically related to the IP argument over the period between the start and end dates.

- Volume(`domain` and/or `IP`, `date`): the total successful lookup volume to the argument domain, IP, or domain and IP tuple on the argument date.

The passive DNS set used in this dissertation began being collected in January 1^{st}, 2011.

Malware Domains We also make use of a separate malware database that contains malware samples and mappings between a malware sample's MD5 sum and binary and the domain names and IP addresses it has queried during dynamic malware analysis. Each entry in the database is a 5-tuple that includes the MD5 of the malware sample, the malicious binary, the queried domain name, the resolved IP address, and the date and time of the analysis. These data are collected from a combination of internal malware analysis output as well as the output from a commercial malware feed.

2.2 Previous Work

Relevant prior work is focused in three primary areas: automatic understanding of malware behavior, understanding malicious infrastructure, and limitations in performing effective APT infrastructure detection.

2.2.1 Understanding Malware Behavior

Deception through gameplay has been discussed [146, 36, 28, 182], or implemented by hand [30], but little empirical work has been done to demonstrate the usefulness such an approach provides. Prior work traditionally focuses on improving information gain generated by honeypots [182, 28] using game theory to model interactions between an attacker and a honeypot operator. Carroll et al. focus on gains generated by having a honeypot masquerade as a normal machine, or vice-versa, and show in which cases a Nash equilibrium can be reached. Wagener et al. similarly tried to achieve equilibrium, but also played games with live intruders. The honeypot was crafted to randomly fail process spawning system calls to coerce an attacker into attempting workarounds for failing tools, hopefully leading to previously unknown tools and exploits. Gaming the botnet C&C network redundancy mechanism, what we refer to as alternative plans, was discussed and used to improve returns generated by a spamming botnet analysis engine [81]. Anticipation games [25], an extension of attack graphs which are based on game theory, were designed to anticipate malicious interaction with a network and determine the answer to questions such as determining the most effective patching strategy for a given network. We differ from previous gameplay work in that we focus on gathering network intelligence, rather than host-level information, and we quantify the usefulness of this network information to security practitioners.

`gza` is similar, but complementary, to other techniques that attempt to coerce malware into revealing useful information. All systems that rely on dynamic binary analysis run into the problem of code coverage, which researchers have addressed by forcing execution of all possible branches [115, 185]. Multipath exploration provides a complete view of possible execution paths of malware but can be evaded with conditional code obfuscation [153] or made impractical due to the exponential explosion in search space. Sharif et. al. describe malware emulators [151], or malware obfuscated by a randomized bytecode instruction set, that would evade multipath exploration. During dynamic analysis, multipath exploration would explore the paths of all possible bytecode programs rather than the execution paths of the malware itself. Since network games do not target binary execution paths, we are resistant to this evasion technique and provide a complementary analysis method. Furthermore, malware increasingly uses external stimuli in the form of *trigger-based behaviors* to determine execution. Malware can determine its execution environment [148, 136, 29] prompting the use of hardware virtualization [45]. More sophisticated techniques include waiting for a specific date to occur or a particular website to be visited. Research has shown how to detect changes in malware behavior as well as determine the underlying cause [10, 23]. We differ from prior work in malware analysis by introducing the concept of evasion-resistant network games. By performing execution path exploration from the network instead of the host, we make it difficult for malware to detect it is being gamed or evade our games.

2.2.2 Understanding Malicious Infrastructure

Prior work has focused on identifying autonomous systems (AS) known to host a disproportionate amount of malicious activity [166, 145, 167]. The idea of network cleanliness [37] has been explored as a potential indicator for future sources of maliciousness based on the assumption that malicious infrastructures tend to group together. We show that, in general, most criminal networks span across multiple autonomous systems, which makes knowing the worst ASs a moot point with respect to performing a comprehensive takedown. Disconnecting an AS from the Internet is not an easy task, and it often does not prevent malicious hosting in the long-term [109]. Focusing on high-level network structures, such as autonomous systems, does not provide sufficient knowledge to perform comprehensive takedowns. In contrast, we focus on identifying the web of smaller-sized networks that work together to provide reliable malicious hosting. Criminal networks that span multiple ASs can be disabled or heavily damaged since we identify not only the malicious networks, but their relationships with others.

On the other end of the spectrum, analysis can be done on individual domains and IP addresses. For example, prior work has studied the infrastructure used to support Rogue AV campaigns [41], fast-flux service networks [86], online scam infrastructure [87], command and control (C&C) networks [31], C&C migration [1], drop-zone infrastructure [72], and pay-per install infrastructure [26]. We consider a campaign to be a collection of domain names and IP addresses that serve a single malicious purpose and are associated with the same threat type, e.g., botnet C&C, drop-zones, etc. These studies provide invaluable insight into the low-level structure of campaigns, but this information also does not suggest how to perform takedowns effectively. The complex structure of criminal networks makes understanding the relationships of the hosting networks essential with regards to takedowns.

Graph-based infrastructure work either represents flows between networks or simply uses the graph abstraction as a way of linking related information. Nagaraja et. al. [118] used game theory and network analysis to suggest effective attacks and defenses against networks and network connectivity. BotGrep [119] identifies botnet communities using random walks to detect dense community structures. Intuitively, peers in a botnet would communicate with patterns distinct from the less structured global Internet. Leontiadis et. al. [94] examined flows from redirections to study the infrastructure used to support illegitimate online prescription drug stores. These approaches all make a simplifying assumption, and treat network structure as simple messaging networks: i.e., two vertices communicating through a connected path in the graph. Christin et. al. [32] built a graph where vertices are domains, bank accounts, and phone numbers and edges are drawn when they appear together in a fraud campaign. This *link analysis* does not follow the typical communication network example, but still yields fruitful results by providing a concrete structure to group related data. Our graph building methodology follows the latter approach in spirit, but also makes use of community finding and network analysis to identify interesting features in the discovered criminal networks.

2.2.3 Existing APT Infrastructure Detection

Industry and academia have proposed APT detection techniques, but we argue `ghost&rae` is the first to do so without also detecting general malicious behavior or relying on non-generalizable signatures for detection. First, we compare `ghost&rae` to existing approaches in industry, followed by comparing against the closest technique discussed in the academic literature: domain name reputation systems.

Many commercial security products claim to perform APT detection, but most rely on signatures and many detect behavior common to most, if not all, malware. Ask et. al. [7] present a literature review of industry techniques that: detect specific malware used by some APTs and requires identifying static malware network signatures [120], rely on static network rulesets [13], detect fast-flux that *may* be used by APT threats [13], and monitor filesystem changes during malware execution [176] (a common behavior in most types of malware). These approaches are not specific to APT [120, 13, 176, 150, 178, 124], rely on signatures and are not generalizeable beyond the malware samples used [120, 13, 176], detect synthetic APTs [124], and are presented with no technical details that we can use to compare against `ghost&rae` [13, 176, 150, 7]. Of these systems shown, only one provided a total of *eight* domains as indicators all of which `ghost&rae` successfully detected; they have been added to our ground truth of over 20 thousand domains.

Domain name reputation systems, such as Notos [4] and EXPOSURE [14] are unlikely to model manual APT behavior, despite the obvious similarities by modeling the properties of DNS resolution and lookup behavior. They assume malicious domains "often look randomly generated and share few common characteristics on the IP infrastructure levels" (zone/network features) and the infrastructure has a history of malicious activity from both malware samples and public blacklists (evidence features). These are not valid assumptions in our case: Section 6.5 show non-random domains. Furthermore, manual APTs use benign, but most importantly, compromised hosts with many years of benign history. Compromised hosts are the primary source of false positives for existing dynamic reputation systems.

CHAPTER III

MALWARE INFRASTRUCTURE ENUMERATION

3.1 Playing Games with Malware

Malware uses the same network protocols that benign software uses when performing malicious activity. Despite the fact that many network protocols exist, nearly all communication on the Internet follows one of two patterns:

1. A transport layer (e.g., TCP and UDP) connection is made to an IP address directly, **or**

2. A DNS query is made for a domain name (e.g., `google.com`) and a connection to the returned IP address is made as in #1.

Higher-level protocols leverage these two use cases for nearly all communication. If we can assume malware relies on these two patterns for contacting its C&C servers and performing its malicious activities, these are the patterns we must target during analysis.

We define a *network game* to be a set of rules that determine when to inject "false network information" into the communication between a running malware executable and the Internet. More specifically, false information is a forged network packet. Consider the running malware sample m in Figure 1(a). Sample m first performs a DNS query to determine the IP address of its C&C server located at `foo.com`. The returned IP address, `a.b.c.d`, is then used to connect to the C&C and the malware has successfully "phoned home". Sample m could also bypass DNS entirely if it were to hardcode the IP address of its C&C and communicate with it directly, as we see in Figure 1(c). This gives us two opportunities to play games with sample m as shown in

Figures 1(b,d): we can say the domain name resolution of `foo.com` was unsuccessful (b) or the direct connection to IP address `a.b.c.d` was unsuccessful (d). At this point, sample m has four possible courses of action:

1. Retry the same domain name or IP address,
2. Remain dormant to evade dynamic analysis and try again later,
3. Give up, or
4. Try a previously unused domain name or IP address.

In (b) and (d), we see the malware samples taking action #4 and querying a *previously unseen* domain name (`bar.com`) and IP address (`e.f.g.c`), respectively. Action #2 is a common problem in dynamic malware analysis systems in general and is further discussed in Section 7.1.

3.1.1 Notation

Stated more formally, let h be a machine infected with a malware sample m that is currently executing in our analysis system running game G_{name}. G_{name} is a packet transformation function called *name*. Given a packet p, $G_{\text{name}}(p)$ represents G_{name} *gaming* p and its value is either the original packet, or some altered packet p' that changes the intent of p. The implementation details of G_{name} determine when to return p or p'. For example, p could contain the resolved IP address of a queried domain name d, whereas p' says d does not exist. In all other ways, such as type of packet and source and destination IP addresses, p and p' are identical. As h communicates with the outside world, it sends question packets, q_i, in the form of domain name queries and requests to initiate a TCP connection and receives response packets, r_j, in the form of domain name resolutions and initiated TCP connections[1]. False information

[1]More accurately, a TCP response packet is a TCP SYN-ACK packet as part of the TCP connection handshake.

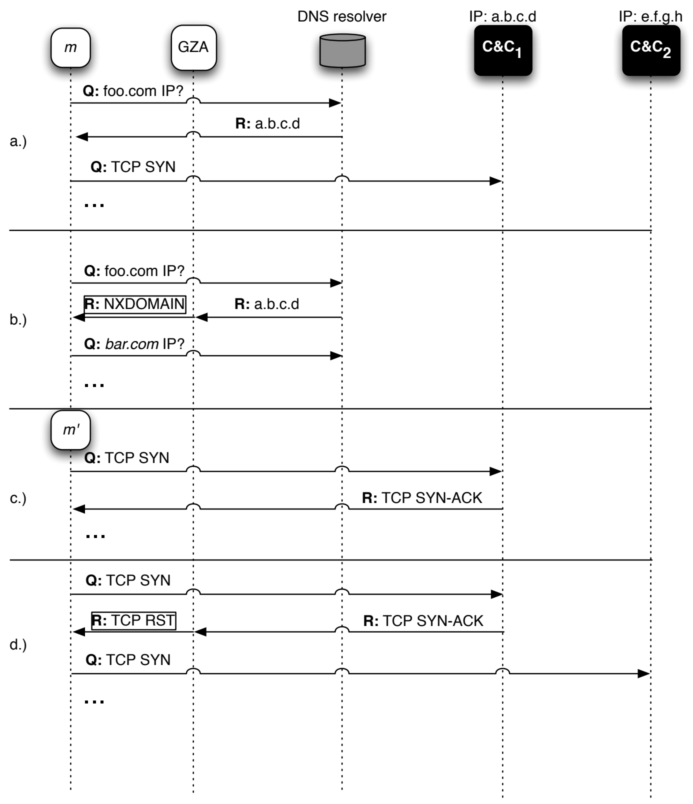


Figure 1: Malware samples m (a-b) and m' (c-d) initiating a connection with the C&C server. m connects by first performing a DNS query to determine the IP address of its C&C server followed by initiating a TCP connection. Sample m' connects directly to the C&C using a hard-coded IP address. Examples (a) and (c) connect without intervention by a game, while (b) and (d) have false information (denoted by boxes) injected.

is provided to the host h by delivering $G_{\text{name}}(r_j)$ in lieu of r_j. A *sample set* for m, $G_{\text{name,m}}$ represents the set of unique domain names and IP addresses queried by m while running under G_{name}. The functions D and I operate on sample sets and return the subset of unique domain names **or** the subset of unique IP addresses, respectively. Given a set of malware sample MD5s, M, a *game set*, G^M_{name} represents the subset of samples that were "successfully gamed" by G_{name}. A game is considered successful if it forces a malware sample to query more network information than under a run without a game present. We formally define this in the following section. The described notation is summarized in Table 1 and will be used throughout the remainder of the paper.

Table 1: Notation for describing games and the sets they generate.

G_{name}	A game called *name*.
$G_{\text{name}}(p)$	The result of G_{name}'s transformation on packet p.
$G_{\text{name,m}}$	The set of network information i.e., unique IP addresses and domain names contacted, generated when malware sample m is gamed by G_{name}.
G^{M}_{name}	The subset of malware samples from M that were successfully gamed by G_{name}.
$D(s)$, $I(s)$	Given a sample set, s, return the subset of unique domain names or IP addresses in s, respectively.

3.1.2 Designing Games for Interrogating Malware

Crafting games without a priori knowledge of malware network behavior is difficult. Furthermore, a successful game for sample m may be unsuccessful for sample m'. By using generic games "en masse", we improve our chances of successfully gaming malware during analysis. We design a suite of games to coerce a given malware sample into showing its alternative plan during analysis. We apply *all* games to a malware sample to improve the likelihood of success. Each game focuses either on DNS or TCP response packets in an attempt to harvest additional C&C domain names or IP addresses, respectively. For a DNS response packet p_d, p'_d is a modified response packet that declares the queried domain name does not exist, i.e., a DNS `rcode` of `NXDOMAIN`. For a TCP response packet p_t, p'_t is a modified response packet that terminates the 3-way TCP handshake, i.e., a TCP-RESET packet. In this paper, we choose to focus on DNS/TCP packets as they are the predominant protocols used to establish and sustain C&C communication; however, our approach is general and can be adapted to other protocols used less commonly in C&C communication. The design of an individual game is based on anecdotal evidence of how malware samples, in general, communicate. We design seven games to perform our analysis of alternative plan behavior in malware:

G_{null} To provide a baseline to compare the effectiveness of future games, this game allows response packets to reach its host without modification. In other words, G_{null} is the identity function.

Note that this does not mean malware communication is allowed to run completely unfettered. We perform standard precautionary measures to prevent malicious activity from harming external systems. However, these measures are not considered part of our network games, but simply good practice when analyzing potentially malicious binaries. We discuss these precautionary measures, which are always performed irrespective of the active game, in detail in Section 3.2.2.

G_{dns1} Malware often immediately connects to its C&C or performs some probing operation to determine the status of its network before doing so. This game assumes the first domain name lookup corresponds to a test of network availability and should be allowed to pass through without modification. Subsequent domain name lookups for domains other than what was queried first will be spoofed. For example, if `google.com` is the first domain queried, all subsequent domains that are not `google.com` will be spoofed. We approximate this behavior in the next game using a whitelist. For a DNS response packet p_d, $G_{\text{dns1}}(p_d)$ returns p_d if its the first DNS request packet and p'_d otherwise. G_{dns1} is successful for m iff $|G_{\text{dns1,m}}| > |D(G_{\text{null,m}})|$.

G_{dnsw} A popular domain name, like `google.com`, is unlikely to operate as a C&C server for a botnet. Therefore, DNS queries on popular domain names are unlikely to be concealing additional malicious network information. For a DNS response packet p_d, $G_{\text{dnsw}}(p_d)$ returns p_d if the domain being queried is whitelisted and p'_d otherwise. Our whitelist is comprised of the top 1000 Alexa domain names [3]. G_{dnsw} is successful for m iff $|G_{\mathrm{dnsw,m}}| > |D(G_{\mathrm{null,m}})|$.

G_{tcpw} An IP address that resides in a known benign network is also unlikely to function as a C&C, much like a popular domain name. For a TCP response packet p_t, $G_{\mathrm{tcpw}}(p_t)$ returns p_t if the IP being queried is whitelisted and p'_t otherwise. Our whitelist is the dnswl IP-based whitelist [51]. G_{tcpw} is successful for m iff $|G_{\mathrm{tcpw,m}}| > |I(G_{\mathrm{null,m}})|$.

G_{tcp1} Malware is often delivered by a *dropper*, a program that downloads, installs and runs the actual malicious binary. If we prevent the dropper from downloading its malicious payload, we will not observe the malicious behavior and fail to unearth alternative plans. We create a class of games that focus on droppers by allowing a variable number of TCP streams to successfully complete before forging response packets. For a TCP response packet p_t, $G_{\mathrm{tcp1}}(p_t)$ returns p_t if the packet is the *first* TCP stream and p'_t otherwise. G_{tcp1} is successful for m iff $|G_{\mathrm{tcp1,m}}| > |I(G_{\mathrm{null,m}})|$.

G_{tcp2} Droppers can have multiple *stages* where malicious payloads are downloaded in more than one TCP stream. This games *two stage* droppers. This game is the same as G_{tcp1}, but allows two TCP streams to complete. For a TCP response packet p_t, $G_{\mathrm{tcp2}}(p_t)$ returns p_t if the packet is the *first or second* TCP stream and p'_t otherwise. G_{tcp2} is successful for m iff $|G_{\mathrm{tcp2,m}}| > |I(G_{\mathrm{null,m}})|$.

G_{tcp3} While one and two stage droppers are fairly common in the wild, we wanted to test for three stage droppers. We can compare the results for G_{tcp1}, G_{tcp2} and G_{tcp3} to determine when we no longer benefit from increasing the number of allowable TCP streams. This game is the same as G_{tcp1}, but allows three TCP streams to complete. For a TCP response packet p_t, $G_{\mathrm{tcp3}}(p_t)$ returns p_t if the packet is the *first, second or third* TCP stream and p'_t otherwise. G_{tcp3} is successful for m iff $|G_{\mathrm{tcp3,m}}| > |I(G_{\mathrm{null,m}})|$.

3.2 `gza`

In this section, we describe the architecture of `gza` and specific implementation details of our system.

3.2.1 Architecture

`gza` contains two components: dynamic malware analysis and gameplay. The first component simply runs malicious code in a fresh virtual machine (VM) and records all network activity that occurs in the VM. All network activity for a VM is routed through one of the games described in Section 3.1.2 as seen in Figure 2. While the malware sample under analysis initiates communication with its C&C, all packets destined for a VM are routed through a network game. The game decides whether to faithfully route the response packet, or construct and send a spoofed packet, to the sample. Games are run on the host machine in isolation from the VMs, so malware cannot detect that its network activity is being analyzed and modified. As discussed earlier, all spoofed responses are RFC-compliant packets of the protocol currently being gamed making them indistinguishable from legitimate responses. As any analysis technique gains traction, malware authors will begin to attempt to circumvent it. Therefore, we discuss possible evasion techniques and how to mitigate them in Section 7.1.

3.2.2 Implementation

`gza`[2] is a collection of Python scripts that run malware samples inside a virtualized Windows XP instance in `kvm` and route packets to implement the games described in Section 3.1.2. All applications and services that could generate DNS or TCP traffic automatically are disabled to ensure that gamed packets are from the analyzed malware only. Before packets are routed for gameplay, precautionary measures are

[2]`https://github.com/ynadji/drop`

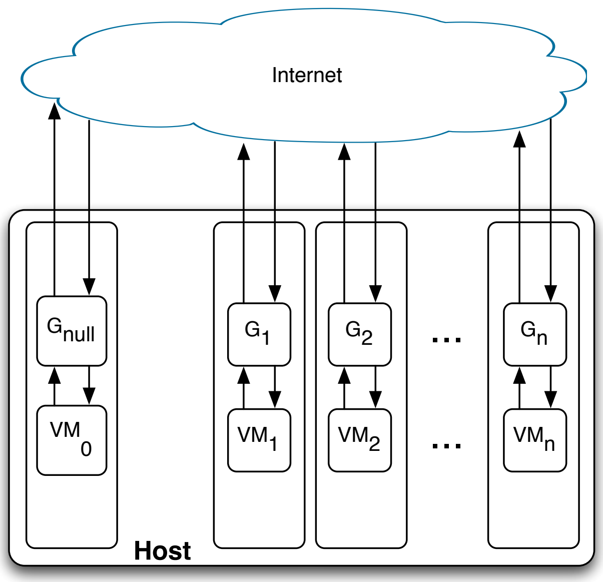


Figure 2: An overview of network traffic routing in `gza`. Multiple virtual machines (VM$_i$) are run on a host using `gza`. Each VM is paired up with a game G and a single sample is run against $n+1$ games. This includes G_{null} to act as a baseline. Each VM's network is isolated from all others to prevent local infection. VM network traffic is routed through its paired game to perform the required packet transformations. There can be multiple groups of these on a single host to perform bulk sample analysis.

performed to prevent malware from damaging external systems. All SMTP traffic is redirected to a spam trap to prevent spamming and traffic to local systems is dropped to prevent local infection of nearby machines or concurrently running VMs. Each VM has its packets routed through the host running `kvm` using `iptables` [121] with relevant packets being forwarded to a Python script that runs a game. This is done through the `iptables` NFQUEUE interface that redirects each packet to a user-mode process which decides if the packet should be accepted or dropped. If the game returns the original packet, the `NF_ACCEPT` message is returned to the host's kernel and the packet is routed faithfully. If the game returns a spoofed packet, `NF_REJECT` is sent to the host's kernel and a forged packet is created and sent to the VM using the packet manipulation library scapy [17].

Games are very short Python scripts that provide two external functions: `playgame` and `spoof`. `playgame` instructs the host's kernel to route the original packet or to drop it; `spoof` generates and sends a falsified packet to the VM if the original packet was dropped. `gza` allows for additional games to be created and removed as its operator sees fit. The implementation of all six of the DNS and TCP games took only 113 lines of code combined.

3.3 Study Methodology

Using the idea of playing games with malware, we design and run three studies to understand alternative plans in malware. The first goal of this study is to understand the prevalence of alternative plans in malware and determine which games are the best in general; successful games force executing malware to reveal the most additional information. The second attempts to quantify how useful this previously unknown information is by determining how long it takes for newly discovered domain names and IP addresses to appear on publicly available blacklists; coerced network information is more useful the longer it takes to appear. The third is to determine if `gza` can be used to understand and detect higher level backup behaviors, such as using a DGA- or P2P-based primary or backup C&C mechanism.

We assume that non-whitelisted network information contacted by malware is malicious. Note that not all games use the whitelist, but during our evaluation we ignore additional network information that is whitelisted. For example, if G_{dns1} caused additional benign domains to be queried, it would not be considered successful. For TCP-based games, we also ignore additional A records returned by DNS requests. We validate this assumption by providing DNS reputation scores for domain names. Furthermore, we show how this increase in network information can improve the accuracy of network-based clustering systems. In both studies presented, all samples are run for five minutes. We discuss timing based evasion further in Section 7.1, but in short the issue is common across all dynamic analysis systems and is orthogonal to the problem we address in this paper.

3.3.1 Representative Study

We created a dataset, D_R, of 2,191 distinct malware samples obtained between April 2010 and October 2010 from a variety of sources, including: low interaction honeypots, web crawlers, mail filters and user submissions. We used several sources to approximate the general malware population as closely as possible. Additionally, all samples in D_R were flagged as malicious by both Symantec and McAfee. For all malware samples $m \in D_R$, we run m in `gza` against each game described in Section 3.1.2. The astute reader will notice that we run the risk of uncovering new information by chance. Consider a malware sample that is analyzed at two distinct times, t and t' where $t < t'$. It is possible that the malicious network infrastructure changed at some time v where $t < v < t'$, which could taint our results. To eliminate this possibility, a single sample is run against all games *at the same time.*

Malware tend to rely on either domain names *or* IP addresses to communicate with their C&C. Using this assumption, we can increase the throughput of `gza` for the long-term study by only using the two most successful games for each protocol.

3.3.2 Long-term Study

In addition to measuring the prevalence of alternative plans in malware, we want to determine how useful this information is the day a malware sample appears on a malware feed. Detecting malicious domain names and IP addresses before they have appeared in blacklists offers a tangible improvement to companies and researchers that use domain name and IP address reputation systems, perform network-based malware clustering, or maintain domain name and IP address blacklists.

Using the two games chosen from the previous study and G_{null}, we play games with malware samples provided by our daily malware feeds over the course of three weeks. Each day, we analyze all the samples we encounter on our feeds using `gza`. Samples that do not generate any network traffic while executing under G_{null} are removed from our results. For each sample that is successfully gamed, we must evaluate the usefulness of the newly obtained information. To do this, we cross-reference the domain name or IP address against eleven public blacklists [40, 99, 48, 50, 102, 163, 97, 43, 133, 140, 132]. The blacklists provide two dates: d_f, the first day a domain name or IP address appeared on the blacklist and d_l, the last day a domain name or IP address appeared on the blacklist. For each additional piece of network information, n_i, and the day it was coerced, t, we place it into one of four categories:

1. `blacklisted`: if n_i appears on any of the blacklists *after* we coerced it on day t i.e., $t < d_f$.

2. `decommissioned`: if n_i appears on any of the blacklists *before* we coerced it but has since been decommissioned i.e., $d_f < d_l < t$.

3. `campaigning`: if n_i appears on any of the blacklists and is *currently* being used i.e., $d_f < t < d_l$.

4. `never`: if n_i does not appear on any of the blacklists.

Each category provides interesting insight into a malware campaign. `blacklisted` network information shows our strategy can coerce domains that other parties eventually flag as malicious. `decommissioned` network information shows that while malware may stop using a network resource, they can quickly and easily resume using one. `campaigning` network information are seen in samples that connect to multiple network resources during *normal* operation. For example, a sample randomly chooses which domain name to use to contact its C&C. `never` network information is perhaps the most interesting. These are domains and IP addresses queried by malware that *never* appear on public blacklists throughout our experiments. `blacklisted` and `never` are the most useful categories of network information and provide the best improvements to systems that rely on such information.

By comparing generated malware sets, we will extract new relationships between malware originally thought to be unrelated. Consider two malware samples, m_1 and m_2 that when run using G_{null} they query domains d_1 and d_2, respectively. However, when run using G_{dnsw}, they *both* query d_1 and d_2. m_1 and m_2 are said to be *strongly related in* G_{dnsw}. Strongly related samples have *distinct* sets of network information when run in G_{null} but *identical* sets when run in any other game. For example, consider a malware family that randomly chooses a C&C domain name to connect to at runtime. MD5 distinct versions of this malware family could have distinct game sets in G_{null} but would have identical game sets in G_{dnsw}. Strongly related samples are likely to be related in some way, for example, they could be members of the same botnet. More formally, two malware samples, m_1 and m_2, are considered strongly related in G_{i} iff:

$$C(G_{\text{i},\text{m}_1}) = C(G_{\text{i},\text{m}_2}) \text{ and } C(G_{\text{null},\text{m}_1}) \neq C(G_{\text{null},\text{m}_2})$$

where C is a function that returns either the subset of unique domain names or unique IP addresses depending on the game type of G_{i} i.e., C is either D or I from Table 1. Samples could be related without being strongly related, however, we focus only on strongly related samples in this paper. Using network games, we can improve malware clustering that uses network features. Furthermore, we use the existing domain name reputation system, Notos [5], to validate that our newly discovered domain names are actually malicious.

3.3.3 Malware Interrogation

In addition to identifying sets of additional domain names or IP addresses revealed during malware execution with gameplay, we can also identify specific classes of backup behaviors using interrogation. By running an individual malware sample under five different execution scenarios, we extract the network endpoints the malware sample used to "phoned home", and based on the differences observed during executions, we identify likely backup plans.

Behaviorally, most malware when presented with unavailable centralized infrastructure resort to one of the following backup plans:

1. The malware simply retries connecting to hardcoded domains and/or IP addresses.

2. The malware attempts to connect to a *finite* set of additional domains or additional IP addresses.

3. The malware attempts to connect to an "*infinite*" set of domains or IPs. This occurs when a malware uses a DGA- or P2P-based backup system.

We can isolate and detect these behaviors by running each sample and applying various packet manipulation scenarios to simulate infrastructure takedown. As a control, we manipulate *none* of the packets during execution. To show that a domain name has been revoked, we run the sample under G_{dnsw}. We run a sample under this scenario twice for durations t and $2t$. To feign IP address takedowns, we use G_{tcpw}. We also run this scenario for durations t and $2t$.

Intuitively, if the number of endpoints (domains or IPs) remains consistent across all runs, the malware sample does not include a contingency plan for C&C failure. If the number of endpoints is greater when the DNS or TCP rewriting is enabled, but remains similar between the two runs with different durations, we expect the malware contains a finite set of additional endpoints as a backup mechanism. However, if we see many more endpoints in the $2t$ duration run than in the t run, this suggests the malware is capable of constantly generating additional candidate domains or IPs to connect to, which indicates DGA or P2P behavior, respectively.

3.4 Analysis

3.4.1 Representative Study

A summary of the results from the representative study are available in Table 2. Of the 2,191 samples in our dataset, 17% were successfully gamed by at least one of the games described in Section 3.1.2. Of the two types of games, DNS-based and TCP-based, G_{dnsw} and G_{tcpw} were successful the most often with 6.0% and 7.5% success rate, respectively. In most cases, the increase in network information was between one and three new domain names or IP addresses for alternative plans. A plot of network information gains is shown in Figure 3. Both graphs are heavily skewed to the right which shows that if a malware author had the foresight to include an alternative plan they used few additional network resources. For increases in IP addresses in Figure 3(a), we see little difference between each individual strategy with respect to the amount of information increase. Figure 3(b) is similarly structured, but with a large spike at 14 additional domain names for G_{dnsw}. D_R contained 37 unique samples of the same malware that all queried the same set of domain names.

In addition to understanding the successes of each game individually, examining cases where multiple games were successful on an individual sample yield insight into understanding malware alternative plans. Table 3 shows this overlap by examining pairwise Jaccard Index [170] of the game sets of each game. The large overlap of 0.93 between $G^{D_R}_{\text{dns1}}$ and the more successful $G^{D_R}_{\text{dnsw}}$ show that a naive whitelisting strategy is sufficient and improves upon hard-coding for common patterns in malware. G_{dnsw} generalizes the behavior captured by G_{dns1}. TCP-based games exhibit a much smaller overlap, primarily due to the specific staged dropper the game targets i.e., G_{tcp2} targets two staged droppers. Game performance dropped from G_{tcpw} to G_{tcp1} and G_{tcp1} to G_{tcp2}. This shows that hardcoding for droppers is less effective than a whitelisting approach.

Furthermore, the small overlap of 0.20 between all DNS-based and TCP-based

Table 2: Summary of results of the representative study of alternative plans in malware. The most successful DNS and TCP strategies are highlighted.

Game	% Gamed	Min Gain	Median Gain	Max Gain
G_{dns1}	4.4%	1	2	28
G_{dnsw}	6.0%	1	3	34
G_{tcpw}	7.5%	1	2	56
G_{tcp1}	6.3%	1	1	54
G_{tcp2}	5.4%	1	1	36
G_{tcp3}	5.4%	1	1	45
Total	17.3%	-	-	-

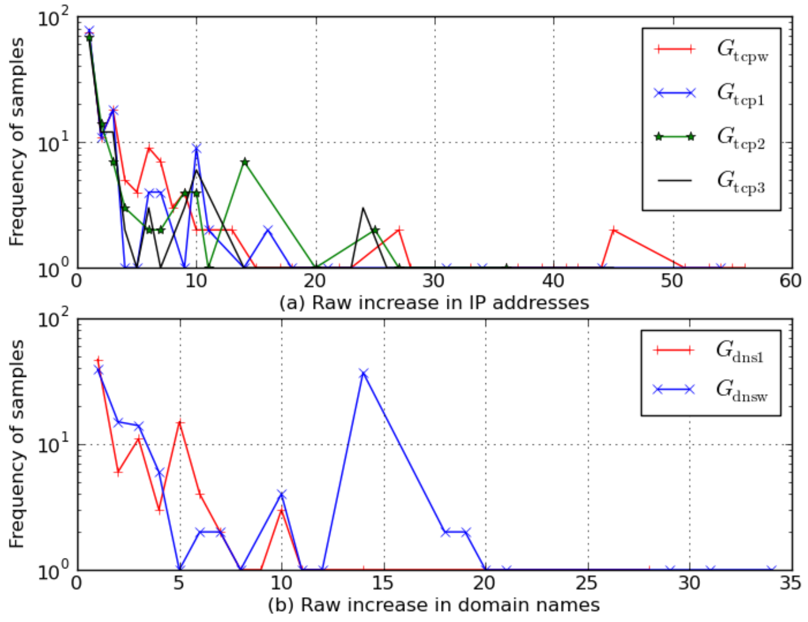


Figure 3: Plot of net network information gains for each game.

gamesets, $G_{\text{dns}}^{D_R}$ and $G_{\text{tcp}}^{D_R}$ shows that malware authors focus primarily on adding reliability using additional domain names or IP addresses for their C&C servers, but rarely both. Since we can approximate our DNS-based games with G_{dnsw} and G_{tcpw} is the best performer among TCP-based games, we will use these two games in our long-term analysis.

3.4.2 Long-term Study

We ran approximately 4,000 malware samples a day through `gza` using three games $\{G_{\text{null}}, G_{\text{dnsw}}, G_{\text{tcpw}}\}$ from March 11th to March 31st. In general, nearly all coerced

Table 3: Overlap in game strategies represented by the Jaccard Index. $G^{D_R}_{\text{dns}}$ and $G^{D_R}_{\text{tcp}}$ are the union of the DNS and TCP game sets, respectively.

	$G^{D_R}_{\text{dns1}}$	$G^{D_R}_{\text{dnsw}}$	$G^{D_R}_{\text{tcpw}}$	$G^{D_R}_{\text{tcp1}}$	$G^{D_R}_{\text{tcp2}}$	$G^{D_R}_{\text{tcp3}}$	$G^{D_R}_{\text{tcp}}$
$G^{D_R}_{\text{dns1}}$	1	**0.93**	-	-	-	-	-
$G^{D_R}_{\text{dnsw}}$	**0.93**	1	-	-	-	-	-
$G^{D_R}_{\text{tcpw}}$	-	-	1	0.50	0.45	0.46	-
$G^{D_R}_{\text{tcp1}}$	-	-	0.50	1	0.36	0.41	-
$G^{D_R}_{\text{tcp2}}$	-	-	0.45	0.36	1	0.43	-
$G^{D_R}_{\text{tcp3}}$	-	-	0.46	0.41	0.43	1	-
$G^{D_R}_{\text{dns}}$	-	-	-	-	-	-	**0.20**

network information was never blacklisted (category `never`) during the course of our study. See Table 4 for an example of the output for a single day of analysis that took place on March 15th. Of the unique domains and IP addresses coerced, approximately 96% and 99% never appear on public blacklists by April 2nd, respectively. A small number were considered `blacklisted`, `decommissioned` and `campaigning`. A breakdown of these categories for the entire study are shown in Figure 4. As shown by the plot, almost all coerced network features never appear on public blacklists.

Table 4: Breakdown of coerced unique network information by category and protocol for March 15th.

Network feature	Category	Count
Domains	Blacklisted	3
Domains	Decommissioned	15
Domains	In Campaign	10
Domains	Never Blacklisted	669
Domains	Total	697
IP	Blacklisted	1
IP	Decommissioned	6
IP	In Campaign	7
IP	Never Blacklisted	3381
IP	Total	3395

The additional network information generated by our games makes relationships between malware samples clearer by providing a more complete picture of C&C communication. Consider a graph K where the vertices are malware samples for a given day of our long-term experiment and edges between vertices represent shared network information. For example, two malware samples that both connect to a domain name

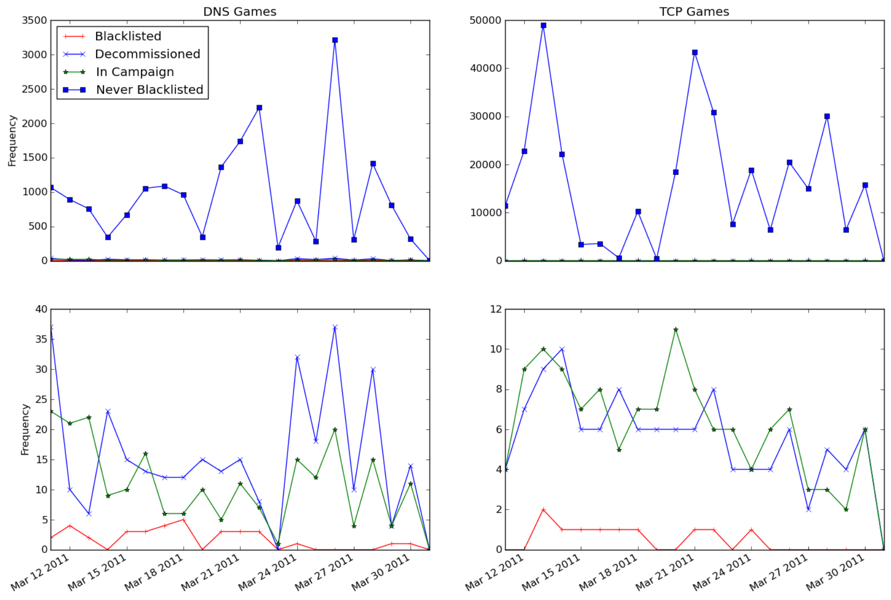


Figure 4: Frequency of network features by category over the course of the entire study. The top row of figures includes all categories, but due to the domination of the `never` category, we also include the other three categories alone on the bottom row. Please note the change in scale.

d would have an edge drawn between them in K. As we uncover more information through gameplay, we add additional edges into K. If these additional edges eventually form strongly related connections between malware samples, we should see a decrease in the number of components in graph K. Figure 5 shows that for G_{dnsw} we always see a drop in the number of components in the game graph of its network information compared to the graph under G_{null}. G_{tcpw} exhibited no change in the graph from G_{null}.

We also used Notos [5] to show the usefulness of our information. Given a domain name, Notos classifies the domain as: suspicious, unknown, or whitelisted. Along with a classification, Notos also provides a confidence score. Notos was trained using four weeks of passive DNS data gathered from six ISP-based DNS recursive sensors located across North America. Notos uses the top 2,000 Alexa 2LD domain names

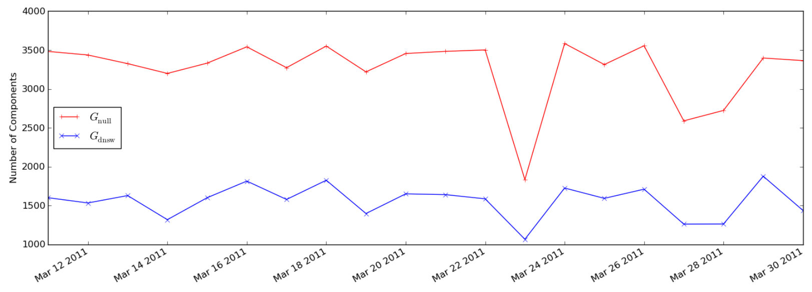


Figure 5: Numbers of components for G_{dnsw} and G_{null} for each day of the long-term experiment.

and the same blacklists used in this study. Of the 161,000 unique domain names contacted during our long-term study, we ran a simple random sample of 15,050 of them through Notos and over 76% of them were flagged as suspicious (see Table 5). The whitelisted domain names were primarily: mail servers, dynamic DNS providers, and content distribution networks. Notos had high confidence in its classification of our coerced domain names: 80% of suspicious domains and 98% of whitelisted domains had confidences above 95%.

Table 5: Domain name classification results and mean confidence values from Notos.

Classification	**Count**	**Percentage**
Suspicious	11,519	76.5%
Not Known	2	$< 0.01\%$
Whitelisted	3,529	23.4%
Classification	**Mean Confidence**	-
Suspicious	0.9731[3]	-
Whitelisted	0.9956[4]	-

3.4.3 Malware Interrogation

We interrogated 591 malware samples from 10 malware families shown in Table 6. The families have known contingency plans with which we can use to tune our heuristic rules to perform the identification. Of the samples analyzed, 433 had no contingency

[3](99% confidence interval $\pm$ 0.0010)
[4](99% confidence interval $\pm$ 0.0009)

Table 6: Malware family training set breakdown for malware interrogation.

Malware Family	**Count**
Expiro.Z	100
Conficker	100
Murofet	97
TDSS/TDL4	92
ZeroAccess	82
zbot	25
Vilsel	25
Onlinegames	25
Fakealert	25
Boonana	20

Table 7: Confusion matrix for malware interrogation.

	dga	**finitedomain**	**finiteip**	**none**	**p2p**
dga	53	1	0	1	0
finitedomain	0	21	0	1	0
finiteip	0	0	0	0	0
none	4	2	1	426	0
p2p	1	1	1	1	77

plan, 55 used a DGA, 81 used P2P communications, and 22 employed a finite set of backup domains. None of the analyzed malware used a finite number of additional IP addresses. Our heuristics successfully classified 97% of the samples' contingency plans correctly. A confusion matrix of the results is shown in Table 7.

This shows that with very simple heuristics one can correctly identify backup behaviors that may spoil an otherwise perfect takedown. We further describe how these results will be used to analyze and assist in performing past and future takedowns, respectively, in Chapter 5.

CHAPTER IV

NETWORK INFRASTRUCTURE ENUMERATION

4.1 Goals and Methodology

Our main objective is to identify the components of network infrastructures used to carry out a variety of criminal activities – such as hosting spam- and phishing-related sites, deploying botnet command-and-control servers, sending spam emails, etc. – and to analyze these malicious network infrastructures to better understand how they are organized and what level of effort would be necessary to take them down. Towards this end, we perform these steps:

1. Enumerate hosts that participate in malicious activities, and find *network relationships* between them.
2. Analyze the structure of these network relationships to identify independent *communities of hosts* that constitute distinct *criminal networks* likely controlled by separate groups of adversaries.
3. Investigate the *criminal network landscape* to identify broad commonalities between classes of criminal networks with respect to remediation strategies.
4. Pinpoint the *critical infrastructure* within a given criminal network that should be targeted during coordinated takedown efforts to increase the likelihood of success, or to *maximize the damage to the adversary.*

To bootstrap the process of enumerating hosts involved in malicious activities and find their relationships, we leverage a large passive DNS database [183], which stores historic records of domain name to IP mappings as observed from live network traffic, and a variety of private and public sources of known malicious domains and IPs (Section 4.1.1). We build an undirected graph where vertices correspond to malicious infrastructure and edges denote a historic relationship between two vertices based on passive DNS evidence. Finally, we apply an analysis based on *community finding* algorithms to identify distinct criminal networks, and we compute the *eigenvector-centrality* of nodes within a criminal network to assess their importance and qualitatively estimate how much potential damage their takedown may cause to the entire criminal network (Section 4.1.3).

4.1.1 Data Sources

To enumerate hosts involved in malicious network activities, we leverage a variety of private and public feeds of domain names and IPs known to have been used for malicious purposes. Since we aim to provide a general picture of criminal networks that may involve different types of criminal activities, we use several sources of information, such as URLs embedded in spam emails, network traces from malware dynamic analysis, lists of known C&C servers, IP blacklists, etc. For example, given a spam URL, we extract the related domain name and use a large passive DNS database to enumerate the set of IP addresses that were recently resolved from this spam-related domain name. Our passive DNS database is constructed from 16 months worth of DNS resolutions collected at a major North American ISP spanning seven different geographical locations and serving several million users.

Our spam feed [77] includes URLs extracted from spam emails captured by a large spam trap. The malware-related data sources are from eleven public blacklists [39, 100, 47, 49, 101, 162, 103, 172, 161, 12, 155] and one commercial malware dynamic analysis feed. The source of information related to C&C servers is an internal company feed comprising domain names and IPs related to known C&C network infrastructures.

To find the network relationships between the enumerated hosts, we leverage two functions that can be defined over passive DNS data:

- *Related historic IPs* (RHIP): given a domain name or set of domain names d, RHIP(d) returns the set of routable IP addresses that d has resolved to at some point in the past.

- *Related historic domains* (RHDN): given an IP address or a set of IP addresses ip, RHDN(ip) returns the set of domain names that have resolved to ip at some point in their history.

Essentially, we consider two hosts to be related if they can be linked via the RHIP and RHDN functions.

After constructing the criminal network graphs, we leverage a commercial threat categorization and attribution process to identify specific criminal operators and malware families that are known to be affiliated with the identified malicious network infrastructures.

4.1.2 Constructing Criminal Network Graphs

In this section, we describe the procedure we use to build our criminal network graphs, which we represent using undirected weighted graphs.

An undirected graph G is defined by its sets of vertices V and edges E. Edges are bi-directional and are assigned a weight between $[0, 1]$ that expresses the "strength" of the relationship between its endpoints. A graph is *complete* if all pairs of vertices are adjacent, and is *connected* if for all pairs of vertices $v_i, v_j \in V$ there exists a sequence of adjacent vertices connecting v_i and v_j. A *disconnected* graph is made up of multiple *components*, or subgraphs of G. If a component contains only one vertex, it is called an *isolated component* [184]. A vertex represents a collection of 256 IP-addresses (a Class C network or /24) and an edge connecting two vertices denotes a historic relationship, according to passive DNS data, between two IPs in the respective Class C networks.

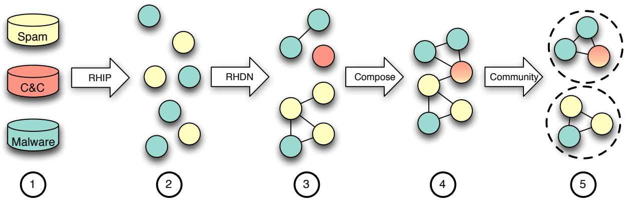


Figure 6: Overview of process to generate criminal network graphs. Data sources are polled (1), domains are converted to IPs (2) and edges are drawn based on overlaps found in the passive DNS database (3). Different source type graphs are composed (4). Graphs are built and composed every day and community finding is performed to identify criminal networks (5).

A high level overview of the criminal network graph generation procedure is shown in Figure 6. Every day, the data sources are polled for new blacklisted network data (1). This network data comes in the form of known malicious IP addresses and domain names. Attackers are known to quickly migrate to new networks after takedowns [109], so in a deployed implementation we keep up with this drift by constantly adding newly discovered malicious network data. All malicious domain names are converted into IP addresses by looking up their related historic IP addresses (RHIP), and all of the IP addresses are binned into the Class C networks (2) that they belong to. Next, we look up each IP addresses' related historic domain names (RHDN) and edges are drawn between vertices when the intersection of their RHDN's is non-empty (3). If network hosts are found to be related to whitelisted domains, these IPs are removed to reduce the occurrences of non-malicious infrastructure in our graphs. Graphs from different sources are composed and edges are redrawn (4). Edges are weighted using the Jaccard index J, a ratio of the cardinalities of the intersection and union of two sets. Given two vertices v_i and v_j that are adjacent, their edge weight is defined by Equation 1,

$$J(v_i, v_j) = \frac{|D(v_i) \cap D(v_j)|}{|D(v_i) \cup D(v_j)|} \tag{1}$$

where $D(v)$ is the set of domains that historically point to IP addresses in vertex v. Graphs from multiple days are composed and community finding is used to identify criminal networks (5).

4.1.2.1 Whitelisting

Our whitelist contains the top 10,000 Alexa domain names and domains of several popular content delivery and advertisement networks. The whitelisting process works by examining the domain name sets generated by RHDN for every IP. Consider an IP *ip*, if its RHDN(*ip*) contains a domain that is whitelisted, or is a sub-domain of a whitelisted domain, we remove *ip* from our graph. For example, consider the domain name `doubleclick.net` which is used by Google's doubleclick advertising service. The top 10,000 Alexa *does not* contain doubleclick.net (only doubleclick.com), however, the IP that doubleclick.net resolves to, 216.73.93.8, has an RHDN set that contains doubleclick.com, which is whitelisted and the IP address 216.73.93.8 would be removed from our graph. If an attacker is aware of our whitelisting strategy there is little room for abuse. For an attacker to abuse our whitelisting strategy to evade our analysis, they would have to commandeer and point a whitelisted domain to their malicious infrastructure.

It is important to stress that we are seeking relationships between IPs as seen from the DNS, *not* from malware samples. For example, a given malware sample may intersperse its connection to its C&C server with spurious lookups to benign domains, these networks will not be connected unless there is an explicit relationship according to our passive DNS database.

4.1.2.2 Community Finding

False positives can still be introduced, despite our whitelisting, which may cause edges to be drawn unnecessarily. For example, if a network host sinkholes multiple domains belonging to distinct criminal networks, our graph building process will erroneously show them as related. To address this problem in general, we leverage graph structure to identify the criminal networks using *community finding* algorithms.

The community finding process can automatically infer these scenarios based on the graph structure and correctly partition the underlying criminal networks. To perform community finding, we use the Louvain method [18], an algorithm known to scale well to graphs with hundreds of millions of vertices and billions of edges. We apply the community finding algorithm to each non-isolated component in our graph at step 5 of Figure 6.

4.1.3 Graph Analysis

Definitions: Understanding whether a graph is dense or sparse is a useful measure for summarizing graph structure. The *density* of a graph G, δ, is defined by $\delta = |E|/\binom{|V|}{2}$ and is the ratio of edges present in G to the number of possible edges in G. A graph with a density of 1 is complete and with a density of 0 has no edges. In our graphs, vertices are not of uniform importance, so quantifying the centrality of a vertex in a graph is a useful way of estimating the node's relative importance in the graph based on its structure. The *eigenvector centrality* (EC) is a measure of a vertex's centrality which often reflects its importance based on the graph's structure. Using EC, a vertex is considered important if it has many neighbors, a few important neighbors, or both. More formally, the eigenvector centrality x_i for a vertex i in a graph G is defined in Equation 2,

$$x_i = \kappa_1^{-1} \sum_j A_{ij} x_j \tag{2}$$

where A is the adjacency matrix of G, κ_1 is its largest eigenvalue, $0 \leq x_i \leq 1$, and x_j are i's neighbors eigenvector centralities [123]. The EC is a useful metric for identifying "important" vertices in a graph independent of the underlying data being represented. We will use this to help determine a takedown strategy that attempts to maximize damage to a criminal network. Removing important vertices targets portions of the criminal network that are used both frequently and collectively to host the operations of multiple criminals.

Consider a social network, such as Facebook, where a vertex represents an individual and an edge drawn between two vertices represents a friendship. Vertices in this graph with high eigenvector centrality will be individuals with a large number of friends, a few friends that have many friends, or both. Similarly, high eigenvector centrality vertices in a criminal network graph are hosting providers that provide redundancy for many smaller hosting providers, a few larger hosting providers, or both. As an example, consider that a botnet operator could host her C&C server using a benign hosting provider, but when the C&C server is discovered, the diligent hosting provider will likely respond to abuse complaints and disable it. Thus, our operator uses a less scrupulous hosting provider to provide redundancy in the event of such a remediation attempt. One can imagine this behavior occurring in several criminals, and aggregated over time one would expect some kind of structure to emerge where the least scrupulous and most diligent hosting providers have the highest and lowest eigenvector centralities, respectively. This intuition suggests that targeting more structurally important vertices can help make takedown attempts more damaging to criminal networks.

There is an important caveat in the social network analogy that concerns connectivity. In a social network, removing social ties can sever friendships between individuals, but the same is not true in criminal networks. This is because nothing flows between connections in a criminal network in a literal sense, like friendship flows between mutual friendships. The assumption that does hold true is that someone with high social standing is likely to befriend additional high status individuals or several individuals en masse. Considering criminal networks, this means high eigenvector centrality networks are more likely to continue and expand their malicious activity into the future and therefore are where remediation efforts ought to be focused.

Simulating Takedowns Our ultimate goal is to determine how to perform effective and damaging takedowns of criminal networks. We first provide a bird's eye view of the criminal network landscape to search for recurring graph structures that are susceptible to takedowns. In other words, graph structures that lend themselves to comprehensive takedowns that require marginal effort. Next, we focus on specific cases of large criminal networks where we identify critical infrastructure to target during remediation to maximize the damage inflicted on a criminal network when a comprehensive takedown is prohibitively expensive.

Using the graph analysis measures we defined above, we identify potential weak points in a criminal network graph that may be susceptible to takedowns, and analyze how successful our takedowns would be by estimating the potential loss in future successful lookups. Not all criminal networks have the same structure, and some structures may be more or less amenable to different types of takedowns, such as taking down specific subnetworks or remediating groups of domain names affiliated with the network.

We consider the two main methods for takedown: *network-level takedown*, accomplished by raiding a hosting facility, or a *domain-level takedown*, accomplished by "revoking" domain names associated with the criminal network in cooperation with the domain names registrars. The goal of these takedown methods is to prevent potential victims from reaching key parts of the criminal network infrastructure.

To determine the order in which to take down infrastructure for a given criminal network G, we define the *criticality* of the vertices $v \in G$ by:

$$crit(v) = v_{ip} \times v_d \times v_{ec} \tag{3}$$

where v_{ip} is the number of malicious IPs within vertex v, v_d is the number of malicious domains that have pointed into v, and v_{ec} is the vertex's eigenvector centrality. The first two measures quantify the vertex's historic career of maliciousness and the eigenvector centrality quantifies the vertex's structural importance to the criminal network.

```
Input: $M_D$: a set of known malicious domains
Output: Returns, for each criminal network, the suggested order of networks
        to eliminate for performing a comprehensive takedown
$M_{\text{IP}} \leftarrow \text{RHIP}(M_D)$
$M_{\text{Net}} \leftarrow$ bin IPs in $M_{\text{IP}}$ into Class C networks
$M_{\text{Net}} \leftarrow \forall_{v \in M_{\text{Net}}}$ remove $v$ if $RHDN(M_{\text{Net}}) \cap$ whitelist $\neq \emptyset$
$E \leftarrow \{\}$
for $v_1, v_2 \in M_{\text{Net}}$ do
    if $RHDN(v_1) \cap RHDN(v_2) \neq \emptyset$ then
        $E \leftarrow E \cup (v_1, v_2)$
    end
end
$G \leftarrow (M_{Net}, E)$
$CriminalNetworks \leftarrow CommunityFinding(G)$
takedowns $\leftarrow \{\}$
for $subgraph \in CriminalNetworks$ do
    takedowns $\leftarrow$ takedowns $\cup$ sort descending by $\arg\max_{v \in subgraph} crit(v)$
end
return $takedowns$
```

Algorithm 1: High-level overview of how criminal networks are discovered and nodes are prioritized for takedown.

In an operational environment, takedowns would be performed based on the output of Algorithm 1. The system takes sets of known malicious domains and outputs, for each identified criminal network, the nodes that should be targeted during a comprehensive takedown to maximize damage to the hosting infrastructure. The infrastructure used by the malicious domains are identified using the passive DNS database call to RHIP. These IPs are pruned using our whitelisting procedure and are grouped into their parent Class C (/24) networks. For each pair of networks, we identify domain name overlaps using the RHDN function. This identifies networks that share the burden of providing malicious infrastucture and if a takedown were desired, must be taken down *simultaneously* to perform a comprehensive takedown. The graph is partitioned using the described community finding algorithm to identify distinct criminal networks and by analyzing the graph structure we can determine which networks provide essential redundant hosting for criminal activity. Because malicious activity is so heavily distributed, targeting the worst individual hosting facility is insufficient. To perform comprehensive takedowns, one must consider the criminal network structure holistically, which motivates the use of the graph-based representation. It allows us to focus on the entire structure such that we can maximize the damage against the network.

For every criminal network in our case study, we order the vertices by their criticality using Equation 3 and estimate the benefit in taking down the criminal network using either network-level or domain-level takedowns. For each type of takedown, we present a cumulative distribution function (CDF) showing the proportion of domain names or networks removed from the criminal network against the total amount of potential victim lookups with respect to the entire criminal network. The intuition is that revoking domain names and blocking IP addresses that received a large volume of queries in the recent past has the potential of preventing a large fraction of the victim population from reaching the criminal network hosts in the future. If we successfully targeted critical infrastructure, the CDF will be superlinear denoting that eliminating key pieces of infrastructure severely impacts the lookups destined for the criminal network. If a strategy is unsuccessful, we should see linear/sublinear CDFs.

4.2 Threat Landscape

In this section, we present general observations about the graphs we built for our study. We discuss source type distributions and describe a case of a frequently occurring graph structure that could be easily taken down.

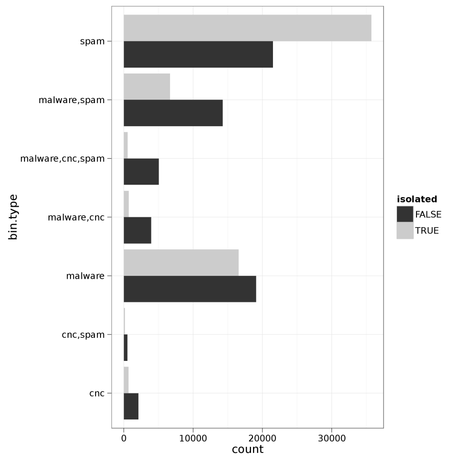
4.2.1 General Graph Statistics

Starting in May 2011, we began building graphs every day for a period of 8 months. Our final graph contains 64,030 vertices and 1,957,614 edges and represents 127,597 malicious IPs and 3,018,077 malicious domain names. The graph is disconnected, where 54% of the vertices are isolated components. These are threats that do not distribute their infrastructure using the DNS. As we mentioned earlier, many of these isolated components may also be due to false positives from non-distributed hosting not present in our whitelist. Figure 7a shows a breakdown of threat types between isolated and non-isolated components. Most isolated vertices hosted spam sites or malware-related threats, and very few hosted any others. Our malware and spam sources are fundamentally noisy which, could explain the large difference between the isolated and non-isolated type distributions.

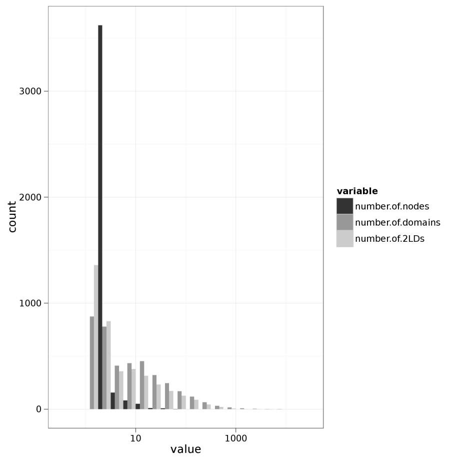
Since we are building our graphs with historical data, it is possible that originally bad IPs are remediated and used later on for legitimate purposes. If the new domains that resolve to the remediated IP space are whitelisted they will be removed from the graph, but if they are not they would still be flagged as malicious. To address this problem in future work, a shorter window of analysis can be used to reduce the likelihood of this behavior becoming commonplace.

4.2.2 Criminal Network Landscape

The remaining vertices form 4,504 distinct communities where each represents a criminal network. Of the 4,504 criminal networks identified, approximately 87% of them formed complete subgraphs. In addition to being complete, Figure 7b shows that



(a) Type breakdown-isolated vs. non-isolated. The y-axis represents the threat type seen in each vertex of our graph. Most host a single threat type (e.g., spam or malware), but many host multiple threat types, even reusing the same IP address (e.g., malware,spam, etc.).



(b) Log-scale distribution of the criminal network size, domains and 2LDs in complete criminal networks.

Figure 7: Threat landscape breakdown

most criminal networks contain few domains and second-level domains (2LD) and even fewer networks. In over half of the complete cases, a criminal network could be disabled by de-registering as few as five domain names or three 2LDs. This strongly suggests that a large number of small criminal networks can be easily remediated.

4.3 Case Studies

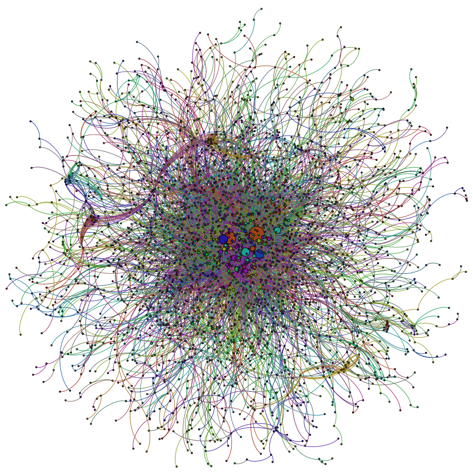
We describe four case studies of large and structurally interesting criminal networks that represent the different classes of infrastructure we saw in the wild. The case studies were not chosen automatically, but were chosen based on the visualizations of the output of our community finding algorithm described in Section 4.1.2.2. We used simple graph metrics to select the case student criminal networks by focusing on large graphs (e.g. many vertices) that had high and low graph densities. In all AS graph visualizations, vertex color encodes the autonomous system number while the vertex size encodes the number of known malicious domains that historically pointed into the network. Furthermore, the edges are drawn when one or more domains are shared between two vertices, unless otherwise specified. In all eigenvector centrality (EC) graph visualizations, vertex shade encodes the eigenvector centrality (darker is more important), and vertex size and edges are defined as they are for AS graphs, unless otherwise specified. The authors suggest that visualizations of the case studies be viewed in a PDF viewer if a high-resolution color printer is not available to get a clear view of the infrastructure.

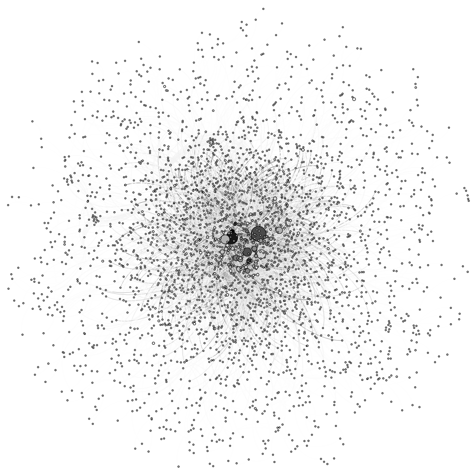
For each criminal network presented, we provide a breakdown of the identified criminal operators using them as well as a breakdown of the sources polled to generate the vertices in the criminal network. Prior to investigating each case study, we were unaware of the underlying criminal affiliations. We will see that EC is a key factor we can use to dynamically obtain a metric for the critical vertices in the criminal network. As we noted in Section 4.1.3, EC is analogous to PageRank [21] for undirected graphs and provides a similar measure of the importance of a vertex in a graph.

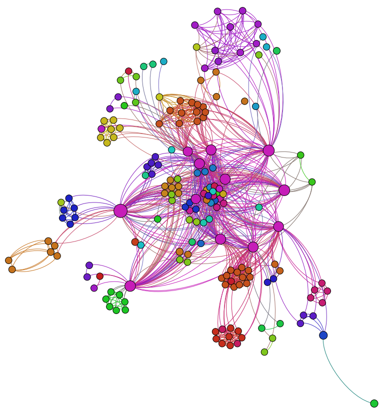
4.3.1 Rustock Criminal Network

Rustock criminal network was among the largest criminal networks we identified with 3,177 vertices and 7,128 edges. Rustock [108] was a large spam-oriented botnet generally used for fraudulent pharmaceutical sales. We describe the malicious hosting infrastructure used by Rustock and that was still in use during our study by other criminals.

Rustock criminal network's most distinguishing features can be seen in Figure 8a. It is sparse (graph density of 0.001) and the graph contains a dense core of networks that contain a large proportion of the domain names compared to the remaining vertices, shown by their larger size. In addition to the number of malicious domains they host, these vertices are also considered important based on their eigenvector centrality, shown in Figure 8b.


(a) Rustock criminal network AS graph


(b) Rustock criminal network EC graph


(c) MojoHost benign hosting network AS graph

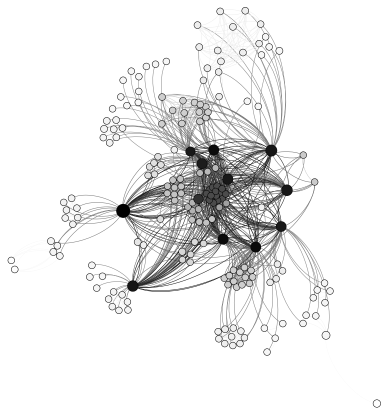
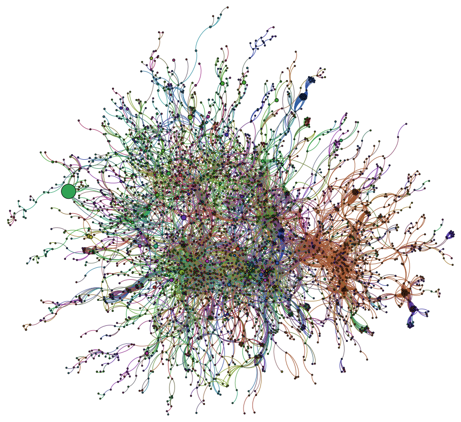
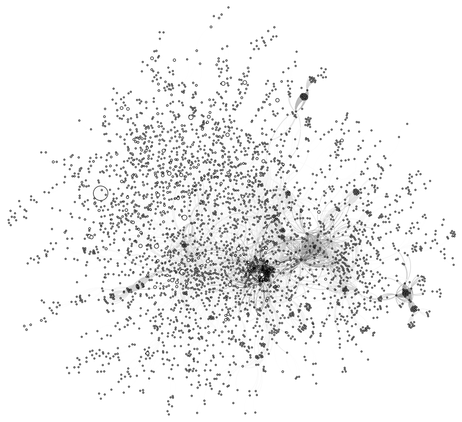
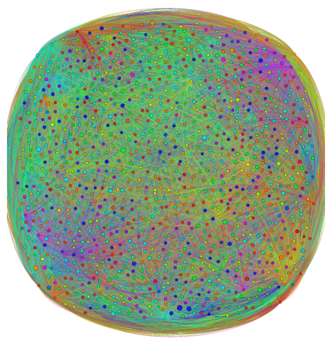

(d) MojoHost benign hosting network EC graph

Figure 8: Case Study Visualizations [11]


(a) Masterhost criminal network AS graph


(b) Masterhost criminal network EC graph


(c) Botnet criminal network AS graph

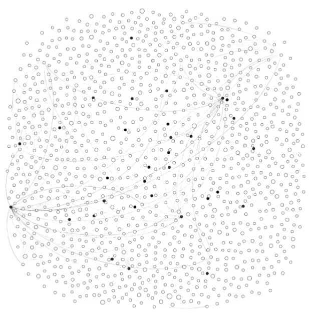
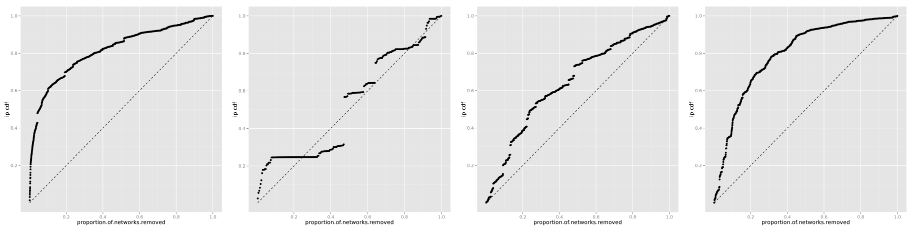

(d) Botnet criminal network **Inverted** EC graph

Figure 9: Case Study Visualizations cont.

Table 8: Top 10 ASes in Rustock criminal network by eigenvector centrality

AS#	**AS Description**	**# of Domains**
33626	Oversee	14,262
22489	Castle Access Inc.	124,321
15146	Cable Bahamas	55,465
13335	CloudFlare Inc.	21,770
16509	Amazon	6,772
32421	Black Lotus Communications	9,070
32592	Hunt Brothers	14,373
21844	The Planet	12,511
26496	GoDaddy	45,654
4635	Confluence Network Inc.	4,635



(a) Rustock criminal network (b) MojoHost benign hosting network (c) Masterhost criminal network (d) Botnet criminal network

Figure 10: Network-level takedown CDFs

The top ASs by eigenvector centrality in the Rustock criminal network are shown in Table 8. This criminal network employs a mixture of bulletproof hosting, cloud-based hosting and compromised home user machines as part of its infrastructure. The inclusion of GoDaddy is due to parking sites the malicious domains pointed to before and/or after their malicious lifetime. CloudFlare is currently running sinkholes for Kelihos and most likely for other botnets as well, which would explain its high importance in this criminal network. Castle Access Inc. and Cable Bahamas are known to be used for domain parking monetization, which would explain their presence.

Rustock was taken down in March of 2011 (Operation b107), however the Rustock criminal network has facilitated other criminal operations until this day. This shows that single botnet takedown approaches can solve only the short term problem of a threat (i.e., spamming activity facilitated by Rustock botnet). In the case of Rustock criminal network, we saw that Internet abuse continued to use the same criminal infrastructure, as the Rustock botnet used to use, long after the botnet was taken off-line. During the 8 months of our experiment, we observed 4,381 new malicious domain names per day that began to use this criminal network.

4.3.2 MojoHost Benign Hosting Network

The MojoHost benign hosting network (Figure 8c) is an example of a benign hosting provider being abused by Internet miscreants for criminal infrastructure. We want to make the distinction clear that we are not saying MojoHost is complicit in criminal activity, but rather, malicious threats abuse MojoHost to build their criminal network. It is a smaller community of 255 vertices that has several distinct campaigns, the "orbiting" sub-communities, using it as infrastructure. The most structurally significant vertices are colored by their eigenvector centrality (Figure 8d). These 12 black vertices all belong to a single AS (AS27589) which provides redundancy for the malicious campaigns.

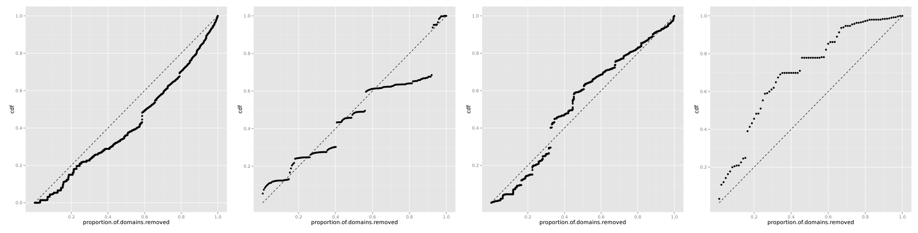
We identified seven distinct operators using the MojoHost benign hosting network for their malicious infrastructure, primarily to act as C&C servers. There were three distinct Zeus kit campaigns, two Blackhole exploit kit campaigns, and three unidentified malware family campaigns running C&C servers. In addition to C&C servers, the community was also home to three data exfiltration drop sites used by a mixture of Zeus instances. The Blackhole exploit kits facilitated drive-by downloads that infected victims with a Delf malware family instance, which is used to perform the second-stage of a two-stage binary drop. Most domains were registered through dynamic DNS providers which are commonly used in Blackhole exploit kit instances.

Despite the fact that the MojoHost community is benign, it presents an interesting hierarchical structure that would intuitively be fairly resistant against AS-level take downs. While the main support structure for the campaigns exists in a single AS, the orbiting communities are spread across 58 ASs in total. If a criminal network contained several layers in this hierarchical fashion, it would be difficult to cripple it quickly due to the redundancy. Maintaining this level of structure may prove to be difficult in scale, which may explain why criminal networks seen in practice are much less organized (Sections 4.3.1 and 4.3.4).

4.3.3 Botnet Criminal Network

This criminal network is a large botnet that provides fast flux services across 1,226 vertices, most of which belong to consumer dynamic IP address space. The graph is almost complete with a graph density of 0.956 (see Figure 9c). It is in the botnet operator's best interest to keep this structure as it maximizes the redundancy of the vertices using DNS agility. Since the graph is nearly complete, it is reasonable to assume that most of the vertices are of about equal importance. The eigenvector centrality, however, reveals interesting underlying structure by highlighting the vertices considered less important to the overall criminal network. In Figure 9d, we see the eigenvector centrality graph where the vertex shading is inverted (darker is **less** important in this case), which highlights 32 vertices within the botnet's sub-structure that are used for other purposes. Specifically, these vertices with lower than normal EC appears to be C&C servers and data exfiltration drop sites for Zeus v2 (a.k.a. Zeus Group B) and Blackhole kit generated malware for a single operator. In this case it is important to note that the only way to truly disable the network is to target the central nodes. Eliminating lower centrality nodes would quickly disable the smaller campaigns contained within, but would not cause damage to the larger criminal network, which is the focus of this paper. Furthermore, significant portion of the domain names in this botnet are related with FakeAV/RogeAV type of threats. One of the main differences of the FakeAV threats facilitated by this criminal network is that they are primarily delivered by search engine optimization poisoning techniques.

Botnet criminal networks are likely to present themselves as dense or complete graphs with a relatively uniform eigenvector centrality distribution due to the fundamental nature of how they are operated by criminals. Furthermore, by looking for vertices that are considered less important by centrality measures, we may identify underlying substructures that differ in function.



(a) Rustock criminal network (b) MojoHost benign hosting network (c) Masterhost criminal network (d) Botnet criminal network

Figure 11: Domain-level takedown CDFs

Table 9: Top 10 ASes in Masterhost criminal network by number of malicious domains

AS#	AS Description	# of Domains
25532	Masterhost	12,281
21788	Network Operations Center Inc.	3,692
3561	Savvis	3,285
7303	Telecom Argentina	2,830
32613	iWeb Technologies	2,684
21740	eNom, Inc.	2,292
25847	ServInt	2,275
16509	Amazon Inc.	2,254
7788	Magma Communications Ltd.	2,225
6939	Hurricane Electric, Inc.	2,201

4.3.4 Masterhost Criminal Network

At 3,725 vertices and 11,519 edges, the Masterhost criminal network is the largest criminal network we identified during our study (Figure 9a). Much like the Rustock criminal network, the Masterhost criminal network is very sparse (graph density of 0.002), but the densely malicious networks are missing from the center. In this criminal network, dense vertices are *not* considered structurally important as shown by Figure 9b. This means that the malicious domains contained within these dense structures are *not* heavily replicated throughout the criminal network, making these good candidates for AS-level takedowns.

The top 10 ASes by number of hosted malicious domains in the Masterhost criminal network are shown in Table 9. Notice the number of domains per AS is substantially smaller than it was for the Rustock criminal network due to the lack of centralized malicious hosting. The biggest AS, with the respect of the domain names that facilitate resolutions for, is the "Masterhost". Masterhost is a very well known bulletproof network that has been identified by the security community since 2007 and it is highly related with the Russian Business Network organization [46]. In the 8 months of our experiments, we observed a median of 1,065 new malicious domain names every day that began to use the Masterhost criminal network.

4.3.5 Simulating Takedowns

Using Equation 3, we identify critical vertices in the case study networks and simulate takedowns by producing the network-level and domain-level takedown CDFs in Figure 10 and Figure 11, respectively. These CDFs show the proportion of networks or domain names removed from the criminal network against the loss in the total amount of potential victim lookups that were made to the entire criminal network. Successful takedowns will manifest as superlinear CDFs, denoting that we can eliminate many potential victims by selectively removing few critical vertices in the criminal network. The aggregate DNS lookup volume to the malicious infrastructure proxies the potential loss in victim population; intuitively, infrastructure that is queried frequently is likely to cause the greatest problems to the attacker if it is taken down. In the two largest cases, the Rustock criminal network and Masterhost criminal network, we see the network-level takedowns are very effective (Figure 10a/10c). In the Rustock criminal network, removing only 20% of the criminal network infrastructure decreases to total number of lookups by 70%. In the Masterhost criminal network, we can decrease total lookups by 40% by focusing our takedown efforts on the worst 20% of the networks. Recall from Figures 8b and 9b that the Rustock criminal network had a dense core of dedicated malicious hosting, while the Masterhost criminal network did not. This would explain the difference in takedown performance between the two criminal networks. Figures 11a and 11c show that domain-level takedowns for these two criminal networks are ineffective, based on the sublinear and linear CDFs. Intuitively, this makes sense as the graphs are very sparse. A single domain name is unlikely to substantially damage the infrastructure because the domain names are less distributed.

Figures 10b and 11b illustrate the difficulty in taking down a well structured network seen in the MojoHost benign hosting network. Since the underlying network infrastructure is benign, the miscreants abusing MojoHost must take great care in distributing their malicious activities, which makes takedowns more difficult. This also suggests that creating hierarchical criminal networks resilient against takedowns is possible, but we did not find these structures in the wild.

For the Botnet criminal network, both network-level (Figure 10d) and domain-level (Figure 11d) takedowns were successful; eliminating 40% of the networks or domains associated with the botnet caused an 80% and 70% decrease in total lookups, respectively. Since Botnet criminal network has a much higher graph density than the other case studies, it makes sense that the domain-level takedown would be effective. However, understanding the success of the network-level takedown requires an understanding of the type of threats the network facilitates the hosting infrastructure for. Most of the malicious hosting that uses the Botnet criminal network are for C&C servers, which need to be highly available. This availability requirement causes the dense structure, which lowers the discriminatory function of the EC metric as most nodes will be considered highly important. Our selection process compensates for this by targeting networks densely populated with malicious domain names and IPs.

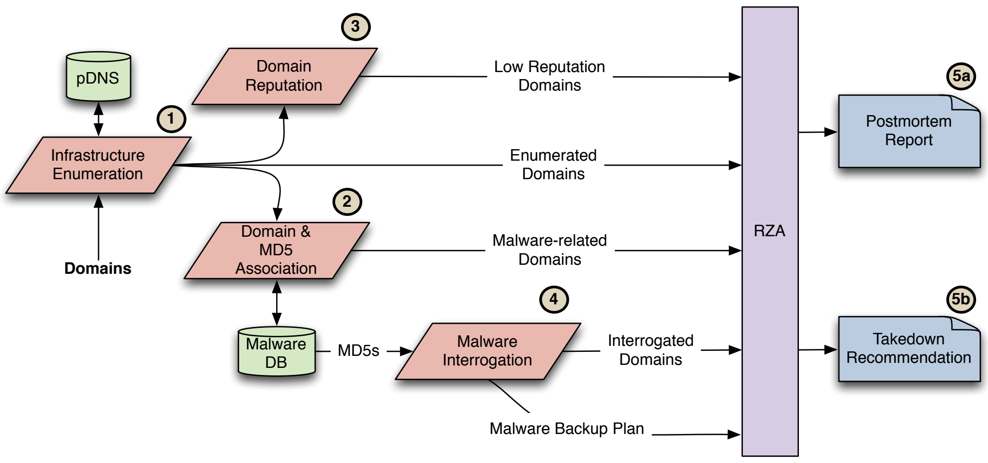


Figure 12: Overview of `rza`.

CHAPTER V

MEASURING & RECOMMENDING TAKEDOWNS

5.1 *rza System*

In this section, we detail the internals of `rza`, our takedown analysis and recommendation system.

5.1.1 Overview

Figure 12 shows the overall process implemented by `rza`. Given a set of known seed botnet domains D_S, `rza` can be asked to generate either a "Postmortem Report" or a "Takedown Recommendation".

In the "Postmortem Report" mode, the input domains represent the domains known to have been targeted by an historic takedown. This produces a report that shows the effectiveness of the takedown of the domain names (Figure 12, step 5a) with respect to the expanded infrastructure `rza` identifies.

In the "Takedown Recommendation" mode, the input domains represent the currently known malicious domains used for C&C infrastructure. Furthermore, the takedown recommendation engine explores possible network resources that may be used by the botnet as a C&C backup mechanism, and suggests any additional measures that must be taken after the primary C&C is disabled to fully eliminate the threat (Figure 12, step 5b).

At a high level, the processing steps executed by `rza` are similar when producing both the "Postmortem Report" and "Takedown Recommendation", despite the difference in inputs and the meaning of the results. The steps are:

1. Expand the initial domain seed set D_S using the pDNS database to identify other domains that are likely related to the botnet's C&C infrastructure. Intuitively, domains are cheap but IP addresses are relatively more expensive. By identifying additional domains that resolve to the same hosts as malicious domains, we can identify other potentially malicious domains related to the botnet.

2. Identify the subset of the expanded domains that are queried by known malware samples. If a domain both points to a host known to facilitate a C&C and is also used by known malware, it increases the likelihood of that domain itself being malicious as well.

3. Identify the subset of the expanded domains with low domain name reputation. Similar to the intuition of Step 2, a domain that points to a known malicious host and also has low domain reputation is more likely to itself be malicious.

4. Analyze the malware samples identified in Step 2. In addition to straightforward dynamic malware analysis, we trick executing malware samples into believing that their primary C&C infrastructure is unavailable using a custom malware analysis system [117] to extract additional C&C domain names. Intuitively,

domains used by malware related to the infrastructure we are studying are likely to be related and malicious. Furthermore, we use the results of the analysis to identify malware contingency plans that would allow the botnet to continue to function after its primary C&C infrastructure has been disabled (e.g., a DGA-based or P2P C&C).

5. Output either the "Postmortem Report" or "Takedown Recommendation" depending on the mode of operation selected at the beginning.

The guiding principle we follow with `rza` is to push our understanding of malicious C&C infrastructure towards completeness. Only once we have fully enumerated the C&C infrastructure can we successfully disable it. We can begin to enumerate C&Cs from the network-level by identifying historic relationships between domain names and hosts using pDNS evidence, and from the host-level by interrogating malware samples. Since the pDNS may contain additional domains not necessarily related to the botnet in question, we identify subsets of domains so we can focus our investigative efforts on those that are most likely to be malicious and not inundate ourselves with information. Each subset serves a different purpose: the low reputation subset holds the domain names from the network-level that are most likely to be malicious. The subset of domains queried by malware represents a reasonable baseline to expect from prior takedowns, as much of this information is readily available to the security community. The subset gleaned from malware analysis contains the domains from the host-level that are the most likely to be malicious. We can use these sets to measure the effectiveness of past takedowns and recommend domains for future takedowns.

In the remainder of this section we describe each of these high-level tasks in detail, and discuss how they work together to suggest a takedown response.

5.1.2 Infrastructure Enumeration

Botnets often make use of the DNS to increase the reliability of their C&C infrastructure, for example using domain name fluxing or simply replacing retired or blacklisted domains with new domains. This cycling of domains, however, leaves a trail in the pDNS database and can be used to enumerate the infrastructure. For example, consider a malware sample m that on day t_1 uses domain d_1 as its primary C&C domain, but on day t_2 switches to domain d_2 to evade the blacklisting of d_1. Assume d_1 and d_2 resolve to the same IP address. Analysis during either t_1 or t_2 yields only one of the possible domains, but the relationship between d_1 and d_2 can be identified in a pDNS database because both resolved to the same IP address.

Using the passive DNS database and the seed domain set D_S, we compute the enumerated infrastructure domain set D_e using Algorithm 2. First, the related-historic IPs (RHIP) of D_S are retrieved and known sinkhole, parking, and private IP addresses are removed. The related-historic domain names (RHDN) for the remaining IPs are retrieved, and any benign domain names are removed, yielding the enumerated infrastructure of D_S: D_e. The relationships retrieved from the pDNS database are within a range of dates to ignore historic relationships that are no longer relevant.

```
**Input**: $D_S$, startdate, enddate: seed domain set, and bounding dates
**Output**: $D_e$: enumerated domain set

$I_b \leftarrow$ set of known sinkhole, private, parking IPs
$W_d \leftarrow$ set of Alexa top 10,000 domain names
$I \leftarrow RHIP(D_S, \text{startdate}, \text{enddate})$
$I \leftarrow I \setminus I_b$
$D_e \leftarrow RHDN(I, \text{startdate}, \text{enddate})$
$D_e \leftarrow D_e \setminus W_d$
**return** $D_e$
```

Algorithm 2: Infrastructure enumeration procedure.

To understand why we filter out benign domains consider an attacker that, in an attempt to mislead our analysis, temporarily has their malicious domains resolve into benign IP space (e.g., Google's) or uses a popular hosting provider (e.g., Amazon AWS). If either of these occur, the D_e domain set may include unrelated, benign domain names. To handle this, we filter domains if they are a member, or are a subdomain of a member, of the set of the Alexa top 10,000 domain names. These domains are unlikely to be persistently malicious and should not be considered for takedown. IP addresses that are non-informative (private, sinkhole, etc.) are also removed, as the domains that resolve to them are unlikely to be related. For example, malware domains sometimes point to private IP addresses (e.g., `127.0.0.1`) when they are not in use, which if not removed would link otherwise unrelated domain names. We use the Alexa top 10,000 in Section 3.3.3, and for consistency we use it here as well. In future work we intend to explore the effect of using smaller and larger whitelists on the generated sets and their accuracy.

5.1.3 Categorizing the Expanded Infrastructure

Not all domains identified during the infrastructure enumeration process are guaranteed to be malicious, but we can identify subsets that are more likely to be malicious. For example, a domain that resolves to an IP address in a virtual web hosting provider is likely to have many benign and unrelated domains that resolve to the same infrastructure as well. To account for this, we focus on domains with known (often public) malware associations, and domains that have low domain name reputation.

Using the passive DNS, we expand the initial seed domain set, D_S, into the expanded set D_e. Next, we identify $D_m \subseteq D_e$ and $D_r \subseteq D_e$, the subset of domain names in D_e with known malware associations and low domain name reputation, respectively. Malware associations are retrieved from our domain name to malware MD5 database and are commonly available in the security community [181]. To determine if a domain name has low reputation, we use a system similar in spirit to [4, 14] which scores domain reputation between 0.0 and 1.0, where 1.0 denotes a low reputation (i.e., likely malicious) domain name. Any domains with > 0.5 reputation are considered malicious and are added to D_r. Unlike D_r and D_m, the set D_i is not necessarily a subset of D_e. Any domains that are used by malware during malware interrogation are added to D_i. These domains expand our coverage as they may unearth domain names that were not previously included in D_e. During our postmortem analysis, we compare these sets to the domains that were actually involved in the takedown (D_S).

Figure 13 shows a Venn diagram representation of a possible configuration of enumerated infrastructure sets. All sets, excluding D_i, are subsets of D_e. D_i is the most likely to include domains outside of the scope of D_e, but suffers the most from the problem of completeness as it relies on dynamic malware analysis.

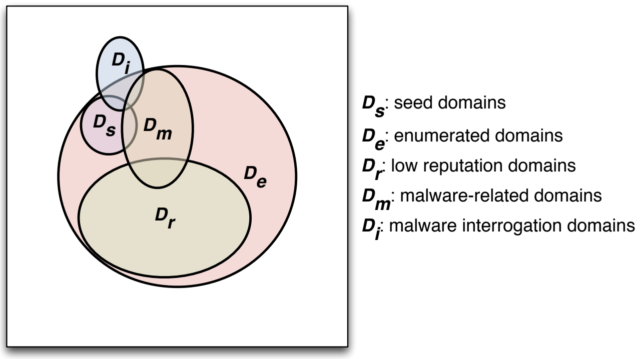

D_i
D_s
D_m
D_e
D_r
D_s: seed domains
D_e: enumerated domains
D_r: low reputation domains
D_m: malware-related domains
D_i: malware interrogation domains

Figure 13: Venn diagram of identified infrastructure sets.

5.1.4 Takedown Recommendation Engine

Using the four aforementioned techniques, we can run our takedown protocol as shown by the decision tree in Figure 14. Suppose we are interested in taking down a hypothetical botnet where the current known infrastructure is $D_S = \{$`01.hans.gruber.com`$\}$. After enumerating the infrastructure, we identify the additional domain name `02.hans.gruber.com` that resolves to the same IP as the `01` child domain. We identify and retrieve the malware samples that have queried the `01...` and `02...` domain names and interrogate them. We identify an additional domain name, `03.hans.gruber.com`, when the first two domain names fail to resolve. Since we identified a finite number of new domain names, we re-run the process with the expanded set of three domain names and this time the malware analysis yields no behavioral changes from what we have already identified. In the event a DGA or a P2P backup scheme is present, the DGA must be reverse-engineered or the P2P network must be subverted as described in [142] after disabling the main C&C infrastructure, respectively.

The question remains which sets of domains should be revoked or sinkholed in order to terminate the botnet's C&C infrastructure, which ultimately must be decided by human operators. In the case where eliminating the botnet is more important than any possible collateral damage that may be incurred, the set of domains in $D_e \cup D_i$ should be targeted, which we consider to be the "nuclear" option. This contains any domain name associated with the C&C infrastructure as well as domains queried by the related malware. In other scenarios, however, this may incur too much collateral damage. We recommend revoking $D_r \cup D_i$ instead in these cases, as these domains are very likely to be malicious. These decisions should be made by threat researchers based on the potential risks associated with deactivating these domain names. Another, less extreme option is to simply block these domains at the network's egress point. This allows enterprise-sized networks to protect themselves while lessening the negative impact incurred by collateral damage.

Ground truth for C&C infrastructure is difficult to come by, which makes evaluating true positives and false positives exceedingly difficult. To roughly estimate this, we present the precision and recall of each set against the "correct" set of $D_r \cup D_i$. If we assume that domains flagged as low reputation or used by malware known to be affiliated with a given botnet are malicious, we can use this union to roughly correspond to ground truth. In our case, the precision of a set D is the fraction of the number of domain names d that are $d \in D \wedge d \in D_r \cup D_i$ over the size of D or $|D|$

and the recall is the fraction between the same number of domain names as in the precision but over the size of the "correct" set, or $|D_r \cup D_i|$.

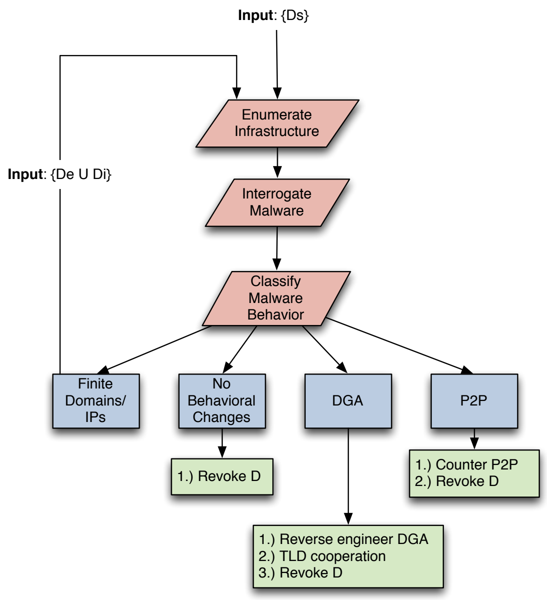


Figure 14: Takedown recommendation engine shown as a decision tree. D in this case represents either $D_r \cup D_i$, which only targets C&C domains that are very likely to be malicious or $D_e \cup D_i$, or the "nuclear" option that should only be used when the threat of the botnet outweighs potential collateral damage.

5.2 *Postmortem Studies*

In this section, we describe how we use `rza` to evaluate historical takedowns. We introduce the takedowns we study and describe the measurements we use to understand the effectiveness of the takedown. We end the section with our experimental results on the postmortem studies.

5.2.1 Postmortem Analysis

For our postmortem analysis, we chose to study the takedowns of Kelihos [111] (aka Operation b79), a Zeus botnet instance [113] (aka Operation b71), and the 3322.org NS takedown that targeted the Nitol botnet [112] (aka Operation b70). We chose these takedowns because they are both recent and high profile. For each takedown, we collect the domains described in the temporary restraining orders (TRO) and use these as our seed domains (D_S).

Measuring Takedown Improvement Prior studies of botnet takedowns relied on secondary measurements, such as global spam volumes, to determine the success of a takedown. Instead, we directly measure the successful domain name resolutions to the identified infrastructure to proxy for the victim population. By comparing the lookup volume to the seed domains (D_S) with the lookup volume to the sets of domains identified by `rza`, we can determine if a takedown was successful and what domains it missed. For example, if all domain sets are equivalent, their lookup volumes will be identical and the takedown would be considered successful.

More formally, for each takedown, t, and its collected seed domains, D_S^t, we generate the enumerated infrastructure sets D_e^t, D_m^t, D_r^t and D_i^t using `rza`. D_e^t is generated using *only* successful DNS resolutions that were issued during the seven days *before* the takedown of t was performed according to the court documents[1]. This allows us to compare what was actually disabled and/or sinkholed during the takedown with what `rza` would have recommended.

For a period of 14 days surrounding the takedown, we plot the successful aggregate daily lookup volume to each of the previously identified sets. To quantify the gains in takedown effectiveness, we calculate the *takedown improvement ratio* as defined by Equation 4.

$$TIR(D_1, D_2) = \frac{MDLV(D_1)}{MDLV(D_2)} \tag{4}$$

Where D_1 and D_2 are two domain name sets and $MDLV$ is a function on domain name sets that computes the median daily successful lookup volume. We use the

[1]These are September 11th, 2012; March 25th, 2012 and September 26th, 2011 for the 3322.org, Zeus and Kelihos takedowns, respectively.

median, rather than the mean, since we are interested in preserving long-term lookup volume trends, which are not captured by outliers. If $TIR(D_m^t, D_S^t) > 1$, this means the subset of D_e of malware-related domain names D_m^t had a stronger lookup volume and accounts for domain names missed by the takedown domains D_S^t. Conversely, if the $TIR \leq 1$, the takedown deactivated related malware domains already and was successful. We also identify malware backup behaviors.

Estimating Risk To provide a different perspective, we also quantify the potential risk of *collateral damage*, or the negative effect of mistakenly taking down benign domains. Ideally, we would represent this by the number of distinct clients that would be denied access to benign services, however, we can once again turn to the lookup volumes to proxy for this.

If we assume all infected botnet hosts behave identically, the aggregate lookup volume on a given day is proportional to the number of infected clients. At most, a single lookup corresponds to a distinct client reaching that domain, however, due to DNS caching effects, differences in malware variant and human behaviors, and network address translation (NAT), this is likely an overestimation of the actual client population. We assume that these behaviors are consistent with respect to queries towards a given botnet.

We quantify the potential risk of collateral damage for a takedown as the difference in the median lookup volume between an enumerated set and the initial seed domain set as defined by Equation 5.

$$Risk(D_1, D_2) = MDLV(D_1) - MDLV(D_2) \tag{5}$$

Using similar notation as seen in Equation 4. Intuitively, the difference between these two quantities is proportional to the number of individuals that would be inconvenienced by this takedown if *all* the domains in D_1 that are not in D_2 *are not* *malicious*. This provides an upper bound on the potential risk involved. The “nuclear option” of taking down all the domains in D_e, or sinkholing all domains that resolve to hosts known to provide C&C for a botnet, is the only way to ensure the C&C communication line is severed, however, this should be weighed against the potential risks.

An analyst wishing to perform a takedown can use the risk values to weigh whether to employ the “nuclear” option or the more reserved options as described in Section 5.1.4. In future work, we hope to improve the risk measure in two ways. First, we can correlate the risk value with the identified true and false positive rates during a real, or simulated, takedown. Furthermore, we wish to more accurately estimate the true population of visitors to infrastructure, malicious or otherwise. This can further help analysts by allowing them to weigh the likelihood of maliciousness against the population that would be affected by a takedown.

For each of the following takedown postmortem analysis, the dashed red line on each plot indicates the date the takedown was performed according to the court proceedings. Each line plot represents the aggregate daily lookup volume to a subset of domains that are either directed to a sinkhole or contained within the enumerated infrastructure sets generated by `rza`. In all cases the D_e lookup volume represents an upper bound of malicious lookups.

5.2.2 Kelihos

The Kelihos botnet was a spam botnet that sent approximately four billion spam messages[114] a day in its first iteration and was targeted for takedown in late 2011. We show the daily volumes for the sets D_S, D_e and D_m for Kelihos in Figure 15. The day Kelihos was taken down, we see lookups to the seed domains completely stop, showing that these domains were effectively remediated. The court order did not specify sinkholes to be used, which explains why the domains simply cease to resolve.

Table 10: TIR and $Risk$ values for Kelihos takedown. These values represent the improvement to a takedown based on `rza`'s output and the potential risk of collateral damage, respectively.

Sets	**TIR value**	**Risk**
D_m, D_S	0.913	-399.5
D_r, D_S	5.690	21,555
D_i, D_S	0.022	-4,492.5
D_e, D_S	10.230	42,415.5

The set of malware-related domains, D_m, and interrogated domains, D_i, also cut off sharply at this point, with a handful of successful resolutions occurring for D_m a few days after the takedown date and ceasing to resolve afterwards. This suggests the initial takedown missed some domains, but these were quickly remediated as well. D_r has a spike similar to D_e, and upon further investigation the spike was revealed to be a malicious domain that resolved into Kelihos' infrastructure but could not be confirmed to be a Kelihos C&C. This domain stopped resolving after the peak date (September 24th).

The computed TIR values are shown in Table 10. Much like the daily volumes figure, the TIR values suggest this takedown was successful. We see large TIR values for D_e and D_r, which indicate additional malicious domains were left unperturbed that resolved into Kelihos' hosting infrastructure. The similar trend between D_e and D_r suggests that many of the extended infrastructure domains are in fact malicious and could have been removed during the Kelihos takedown effort.

For the D_e and D_m sets, we have precision and recall of 0.22/0.67 and 0.25/0.03, respectively. The recall for D_e is quite low as this means upwards of 30% of the domain names that are likely malicious were harvested from malware interrogation. This stresses the importance of labeling both from network information from pDNS, as well as information gathered from malware.

According to the analysis by `rza`, this takedown was largely a success, however, we know that new variants of Kelihos emerged soon after. Analyzing its 168 malware samples from before the takedown shows that a P2P C&C mechanism existed as a backup plan in the malware, which may have helped bootstrap its resurgence. This stresses the importance of being prepared to counter malware behavior after its primary infrastructure has been disabled.

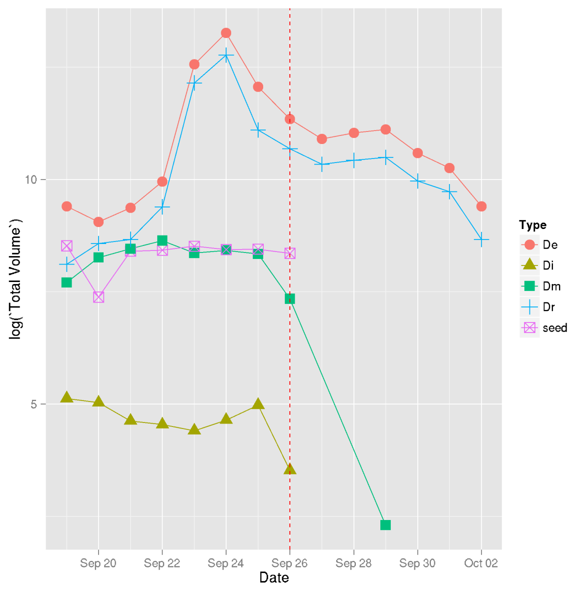


Figure 15: Kelihos aggregate daily lookup volume (log-scale).

5.2.3 Zeus

The Zeus takedown targeted a large botnet that used the popular malware kit Zeus to create its malware. This takedown relied on sinkholing the seed domains. We show the daily volumes for the sets D_S, D_e, D_m, D_r, as well the volumes for *domains in* D_e that resolve into sinkholes operated by Microsoft and the other sinkholing party, in Figure 16. Of the 2,825 malware samples analyzed, none of them included a P2P- or DGA-based contingency plan.

The first observation is that unlike in the case of Kelihos, Microsoft began sinkholing domains *before* the date specified in the court order as evidenced by the non-zero query volume resolving into Microsoft's sinkholes before the takedown date. To reiterate, domains that resolved only to the sinkhole before the takedown date were not included to prevent prior uses of the sinkhole from interfering with our results. Furthermore, the volume of lookups that resolve into the sinkhole are orders of magnitude larger than the lookups only to the seed domains, suggesting that domains *not* specified in the court order were also sinkholed. We see a spike in lookup traffic directed towards the seed domains and domains that resolve to Microsoft's sinkhole, indicating increased sinkholing action at the time of takedown. `rza`'s D_m set captured fewer domains than those sunk by Microsoft's sinkhole, however, there is a large discrepancy in lookups to domains flagged as malicious by our reputation system, i.e., lookups to the domains in set D_r. We see a drop in lookups to D_r that corresponds to the Microsoft sunk domains, which indicates D_r subsumes the set of sunk domains. The other sinkhole operation experienced a similar drop after the Microsoft takedown, which suggests there was contention over which domains belonged to which sinkhole.

In this takedown, the ad-hoc nature of takedowns made coordination between companies difficult and the lack of oversight allowed the court order to not be followed exactly. While Microsoft was clearly sinkholing more domain names, the takedown interfered with an existing takedown. Without a centralized method of communicating who is sinkholing what, this pattern of stepping on other researchers' toes is likely to continue.

The computed TIR values are shown in Table 11. We compare against both the seed domain set, and the set of domains resolving into Microsoft's sinkhole. With respect to the seed domain set, we nearly tie considering malware-related domains and capture many more lookups to potentially malicious domains when considering the dataset derived from reputation, D_r. The story is similar when compared to domains that resolve into the Microsoft sinkhole, but to a lesser extent. Recall the volumes for Microsoft sinkhole resolutions *only* include domains we identified in D_e. This suggests that not only were these deemed malicious by a 3rd party, but they were added by Microsoft independent of the domains listed in the court order.

Table 11: TIR and $Risk$ values for Zeus takedown.

Sets	**TIR value**	**Risk**
D_m, D_S	0.979	-11,357.5
D_r, D_S	3.921	1,641,580
D_i, D_S	0.148	-478,874
D_e, D_S	14.321	7,486,221
D_m, D_{mssink}	0.553	-444,265.5
D_r, D_{mssink}	2.215	1,208,672
D_i, D_{mssink}	0.084	-911,782
D_e, D_{mssink}	8.091	7,053,313

For the D_e and D_m sets, we have precision and recall of 0.03/0.98 and 0.30/0.01, respectively. Most of these values are quite low, with the exception of D_e's recall, which is unsurprising. This indicates most of the malicious domains could be identified through passive DNS. The low precision value for D_e indicates that many of these domains should probably not be targeted in a takedown and the low precision for D_m suggests that while many have low reputation and are likely malicious there are no known malware associations, reinforcing the motivation for using domain name reputation.

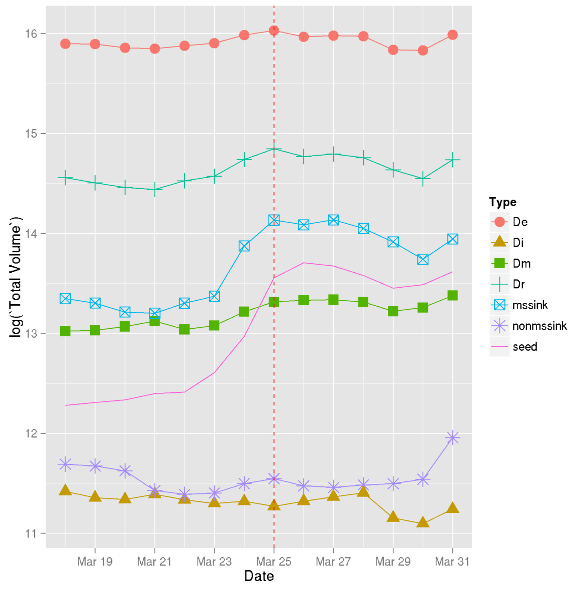


Figure 16: Zeus aggregate daily lookup volume (log-scale).

5.2.4 3322.org

The 3322.org takedown represents the most extreme case where `rza` would have improved a takedown's effectiveness. This takedown was accomplished by transferring the entire 3322.org Name Server's (NS) authority to Microsoft and domains deemed malicious resolved to a set of known sinkhole IP addresses. The daily volume plot for 3322.org is shown in Figure 17. Unlike the Zeus takedown, domains were sunk on the day of the takedown and were limited to *.3322.org domain names. Unfortunately, this only accounted for a fraction of the lookups to domains with known malware associations, D_m, and domains with low reputation, D_r that resolved to hosts known to support malicious activity. We notice a drop in lookups to D_m and D_r when the takedown is performed, showing that most of the domains targeted by the takedown were likely malicious, however, the lookups to remaining infrastructure identified by `rza` are still frequent. We see D_i closely matches the sinkholed domain names, suggesting this is the primary method that was used to identify the takedown domains. Unlike the previous two cases, all enumerated sets have TIR values greater than one. This agreement suggests that malicious domains were almost certainly missed during the 3322.org takedown effort. Of the 10,135 malware samples we analyzed, none of them had a P2P- or DGA-based contingency plan.

This case shows the importance of using multiple sources to determine related malicious infrastructure before performing a takedown. Simply identifying domains with known malware associations offers a substantial improvement on the effectiveness of the takedown. Further, the similarity between the D_m and D_r trends shows most of the domains overlap between the two, which only further bolsters the likelihood that they are indeed malicious. To make matters worse, all the domains that were not sinkholed were given enterprise-level domain name resolution services, despite the high probability they were involved in malicious activities. The computed TIR values for the 3322.org takedown are shown in Table 12. Unlike the previous two

Table 12: TIR and $Risk$ values for 3322.org takedown.

Sets	TIR value	Risk
D_m, D_{mssink}	13.821	409,593.5
D_r, D_{mssink}	18.956	573,627.5
D_i, D_{mssink}	1.049	1,560
D_e, D_{mssink}	654.940	20,890,774

postmortems, `rza` identified numerous additional malicious domains that were left undisturbed by the takedown on 3322.org.

For the D_e and D_m sets, we have precision and recall of 0.06/0.95 and 0.38/0.03, respectively. These results are similar to those for Zeus and further reinforce the need to include domain reputation as a measure in `rza`. Simply relying on passive DNS (for D_e) and malware associations (for D_m) overestimate and underestimate the malicious domain names, respectively.

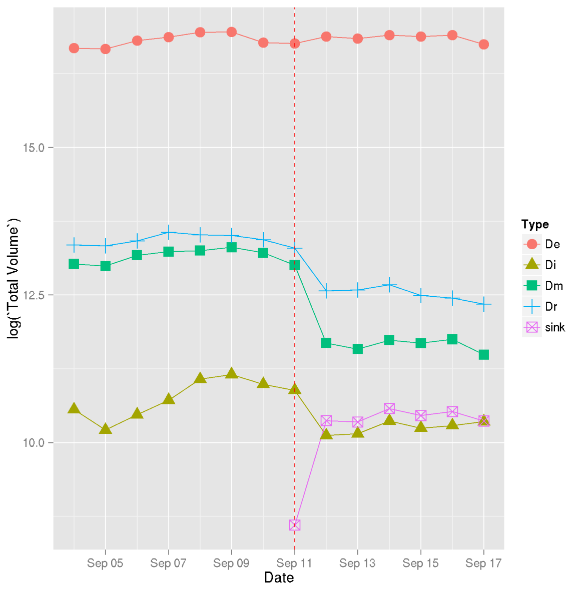


Figure 17: 3322.org aggregate daily lookup volume (log-scale).

5.3 Takedown Recommendation Analysis

In this section, we run `rza`'s takedown protocol on 45 botnet C&Cs being tracked by Damballa, Inc. during the month of April, 2013 and present the results. We chose to use the C&Cs already tracked by Damballa out of convenience and it is important to stress that they could be substituted by any set of domain names known to correspond to a botnet's C&C infrastructure. There are many publicly available sources of this information that allow similar experiments to be repeated. The calculated TIR values and predicated backup plans for the 45 botnets are shown in Table 13 and the associated Risk values are shown in Table 14.

Overall, the TIR values indicate we are gaining additional infrastructure information as we did in the postmortem cases. Similarly, we see very large TIRs for the expanded infrastructure set, D_e, further evidence that infrastructure related by the passive DNS includes additional domains that need to be narrowed down by categorizing the domain sets. These TIR values were smaller, as we saw with the postmortems, with the malware-related and reputation-based sets—D_m and D_r, respectively—contributing the bulk of the newly observed lookup volumes.

In addition to describing the enumerated infrastructure sets, we also identify the backup mechanisms, if any, present in the botnet. If a botnet's malware has no backup plan, it is a prime candidate for a smooth, DNS-only takedown, otherwise we have identified the necessary conditions for performing an effective takedown. The most important finding, however, is that of the 45 botnets we studied 42 of them had no contingency plan for central C&C failure, suggesting the bulk of these botnets can be successfully taken down *without* requiring additional measures, such as reverse engineering a DGA or combating a P2P-based C&C. This suggests that while performing a takedown is difficult, we are likely to succeed in many cases.

Table 13: Recent botnet TIR values (compared against D_S) and backup plan classification.

ID	$\mathbf{D_e}$	$\mathbf{D_m}$	$\mathbf{D_r}$	$\mathbf{D_i}$	**Backup Plan**
1	7.229	2.719	6.815	1.446	finite-domain
2	13.669	0.000	23.891	275.751	p2p
3	0.856	0.000	0.764	0.142	none
4	2.808	1.158	2.602	0.554	dga
5	12.005	10.117	11.612	0.023	none
6	20.632	1.448	15.665	0.044	none
7	2.130	0.015	0.798	11.917	none
8	289.387	154.932	233.521	0.000	none
9	42.570	0.000	23.522	1.395	finite-domain
10	0.746	0.000	0.597	0.241	finite-domain
11	3.783	1.068	3.208	0.255	none
12	13.115	3.809	11.896	0.246	none
13	10.139	1.698	8.726	0.697	none
14	8.266	0.000	8.259	3.190	none
15	2.028	0.189	0.131	0.094	finite-domain
16	471.226	76.176	392.788	1.045	finite-domain
17	87.004	33.807	72.759	4.052	none
18	27.036	1.810	14.952	1.021	none
19	8715.005	816.740	8696.197	8.520	none
20	170.752	0.743	10.210	0.405	finite-domain
21	7.260	2.012	5.828	0.056	none
22	13.492	1.364	12.011	0.000	none
23	14.146	0.000	12.891	7.449	none
24	52.593	0.000	52.174	0.967	finite-domain
25	223.201	6.869	21.504	0.000	none
26	1067.604	0.000	1062.696	0.001	finite-domain
27	293.466	46.070	232.437	0.005	none
28	1.251	0.311	0.923	0.082	dga
29	13.589	0.547	4.886	1.100	finite-domain
30	380.568	27.986	350.686	0.082	none
31	17.700	0.000	16.837	0.500	finite-domain
32	5.857	4.679	5.724	0.575	finite-domain
33	25.042	4.085	21.460	3.094	none
34	0.156	0.139	0.152	0.014	finite-domain
35	25.048	1.272	15.638	0.000	none
36	12.121	7.364	11.183	0.336	finite-domain
37	11.698	10.329	11.544	2.321	none
38	91.364	0.457	9.618	0.000	none
39	4.640	1.085	3.694	0.599	finite-domain
40	7.491	0.303	6.865	257.059	finite-domain
41	3.161	0.485	2.700	0.187	finite-domain
42	2.378	0.487	2.372	1.288	finite-domain
43	33.227	12.958	31.650	3.014	none
44	21.761	2.219	3.061	1.108	none
45	2.217	0.101	2.150	0.103	none

Table 14: Recent botnet *Risk* values (compared against D_S).

ID	$\mathbf{D_e}$	$\mathbf{D_m}$	$\mathbf{D_r}$	$\mathbf{D_i}$
1	376,591	103,904	351,529	26,974
2	6260	−494	11,312	135,766
3	−7425	−51,681	−12,205	−44,352
4	694,233	60,473	614,941	−171,099
5	485,427	402,158	468,070	−44,110
6	362,341	8262	270,663	−18,457
7	6103	−5320	−1088	58,936
8	1,249,414	666,902	1,007,383	−4332
9	255,237	−6140	138,285	2427
10	−5122	−20,195	−8131	−15,320
11	3,024,316	73,348	2,399,494	−1,086,629
12	89,727	20,801	80,695	−7406
13	9,150,880	698,974	7,735,834	−1,001,304
14	17,959	−2472	17,942	5412
15	585	−461	−494	−515
16	4,106,017	656,434	3,421,089	−8732
17	80,537	30,721	67,197	−936
18	3128	97	1676	−120
19	277,603	25,987	277,004	−32
20	5,719,294	−8674	310,317	−33,692
21	1,061,289	171,489	818,507	−169,516
22	5,586,161	162,678	4,923,961	−447,172
23	478,741	−36,417	433,040	234,848
24	204,816	−3970	203,153	−130
25	9,034,976	238,626	833,715	−40,652
26	31,892,669	−29,901	31,745,933	−29,860
27	2,023,910	311,893	1,601,580	−6889
28	692	−1897	−212	−2528
29	26,164,386	−942,098	8,075,579	−2,078,370
30	341,286	24,265	314,418	−825
31	23,614	−1414	22,393	−707
32	392,925	297,607	382,154	−34,345
33	3,798,382	487,342	3,232,483	−157,989
34	−6385	−6519	−6416	−7466
35	468,361	5289	285,099	−19,476
36	741,041	424,070	678,530	−44,278
37	452,686	394,741	446,180	−42,314
38	6,688,344	−40,216	637,882	−74,015
39	1,676,605	39,173	1,240,928	−184,676
40	5,515,527	−592,194	4,983,178	217,562,247
41	160,793	−38,321	126,529	−60,522
42	224,265	−83,469	223,419	46,946
43	243,983	90,534	232,045	−7571
44	3,648,885	214,216	362,154	−175,760
45	498,000	−367,626	470,334	−409,148

CHAPTER VI

COLLATERAL DAMAGE—ADVANCED THREATS

One of the most difficult aspects of performing a takedown is identifying and avoiding potential collateral damage, as noted in past takedowns [62]. While there are many kinds of collateral damage, we focus on identifying a particularly heinous kind: advanced persistent threat (APT) infrastructure. Not only are APTs a serious threat to be handled in their own right, they frequently use compromised machines to support their infrastructure, both of which makes them ideal targets for takedown as well as increased care due to the possibility of disabling legitimate infrastructure in the process.

6.1 What We Know About APT

Threats focused on espionage or intellectual property theft are difficult to automate beyond exploitation or lateral movement within the target network. Intuitively, it is difficult to know exactly what intellectual property ought to be stolen from a network, how the data are labeled, and where they are located. The manual nature of such operations is illustrated by C&C monitoring occurring during business hours in the perpetrator's country [42] and in sleeping behavior [165, 104] seen in the C&C infrastructure. For example, "domain names used ... only point to actual control infrastructure during very short time windows" and resolve to "parked" or local/private IP space (e.g., 127.0.0.1, 0.0.0.0, etc.) [165] when not in use. This lowers the profile of the malware in the target's network and makes it far less detectable by conventional means.

When a threat actor wishes to explore the network, she changes the domain name to resolve to an active host and the infected machines phone home to provide remote access. After the exploration is complete, the domain name is once again parked to a local/private IP address. This manual behavior is entrenched such that “when they need to change the IP resolution of [a domain], they simply log in to these services and update the IP resolution of their [domain] via a web-based interface” [104], further reinforcing the manual efforts that take place in infrastructure management. While automating the process of updating a domain name resolution is trivial, knowing exactly when an attacker wishes to poke around a victim network is not.

Most of the APT reports we studied [84, 105, 104, 165, 95, 61, 64, 137] rely on domain names for C&C communication. Domain names are cheap (or free), easy to use, and employed by the run-of-the-mill remote access tools (RAT), a common tool for APT actors, making them a natural choice for infrastructure. Some threats [65] do not use the DNS, but these are less agile, easier to detect, and are not the focus of this paper. While it is possible to design C&Cs that operate over covert channels [88], bypassing the DNS entirely, we consider this outside of the scope of this paper as such an APT has never been uncovered.

APTs commonly make use of compromised machines, or “stepping stones,” to route traffic as “camouflage for their attacks” [104]. Many reports [84, 105, 104, 165, 95, 61, 64] confirm the use of stepping stones in APT C&C infrastructure and suggest in many cases they are compromised machines. Using compromised machines for infrastructure has two primary benefits: C&C infrastructure can be “near” the infection site so that alarms will not be raised, and compromised infrastructure does not leave behind a money trail, giving plausible deniability to the APT actors. In either case, we expect to see *both* benign and APT-related domain names on a given host used for APT C&C infrastructure. In fact, some APT instances *required* compromised machines to achieve their ends. In one APT that targeted the financial sector, IP whitelisting at the targeted bank's edge network necessitated compromising a whitelisted machine to act as a proxy for the C&C to be tunneled through [105].

Lessons Learned Advanced persistent threats (APT) are vague and differentiating them in a reliable, automated fashion from traditional threats and malware is difficult. Based on reports from industry, we focus on a class of APTs, called "manual APTs", that are: *manually operated*, *primarily DNS-based*, and make use of *compromised machines* to proxy command and control (C&C) messages.

It is possible for APT actors to evade these assumptions, but it comes at a great cost to the operators. By not relying on traditional infrastructure, they evade potential attribution pitfalls. In the case of APT threats, simply being able to reliably pin the origin of an attack to a country or organization is enough to cause geopolitical strain, which is not the case for traditional botnets. One must only look to the political issues between the United States and China to see the potential downfall to APT attribution. It is in the best interest of the attackers to lie low and maintain plausible deniability. Relying on carefully maintained, compromised assets in lieu of traditional infrastructure is paramount to successful APT operations. We focus on these common properties in the threats we studied to derive the features we use for modeling in `ghost&rae`.

6.2 System Overview

Our system, `ghost&rae`, is composed of three logical components: a data collection component, a supervised learning component, and an unsupervised learning component. An overview of the entire system is presented in Figure 18.

The first step for the system is to collect various datasets (i.e., passive DNS, BGP routes, APT reports) and properly store them in a distributed file system. Once this operation is complete, the dataset becomes available to the two learning components of the system (Step 1, Figure 18). The two data learning components work in concert

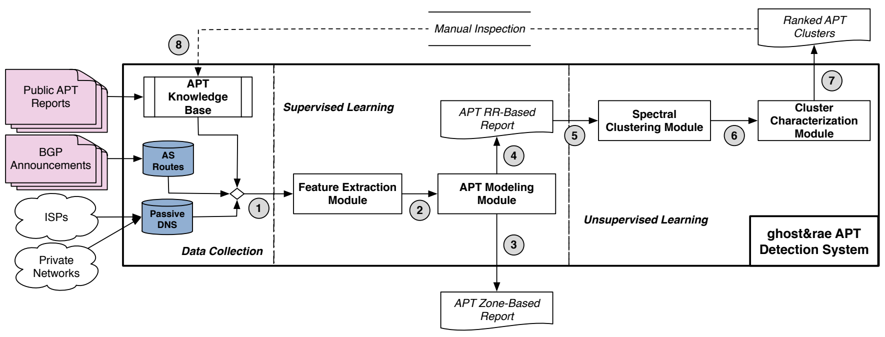


Figure 18: Overview of `ghost&rae` APT detection system.

to provide reports to knowledgeable network operators to improve their ability to respond to APT threats in their networks.

The supervised component (Steps 2-4, Figure 18) classifies resource records (RRs) as being APT or non-APT related. It can also convict entire domain name zones by taking a majority vote of the RRs under a zone. For example, if `ftp.evil.com`, `www.evil.com`, `pop.evil.com` are domains that are classified as APT, APT and non-APT, respectively, and are the only subdomains of `evil.com` seen in our passive DNS database, the system would convict the entire `evil.com` zone as APT-related. In addition to simple classifications, the supervised component also provides a confidence score that denotes how likely it is that the RR belongs to one of the two classes. This is used to improve the unsupervised component.

The unsupervised component in `ghost&rae` comprises Steps 5-7 in Figure 18. At a high-level, it clusters together related APT infrastructures by observing the relationships between domain names and IP addresses. Not only does this make evaluation and report generation easier, it also allows us to group similar APT campaigns. Given our assumptions, we expect a given IP address to be resolved by both APT and non-APT domain names, so we must inform the unsupervised component how to distinguish between these two classes and perform a clean separation between targeted attacks. Using the classification output from the APT modeling module, we weight these relations based on their likelihood of maliciousness to distinguish between APT and non-APT domain names when clustering. Clusters are ranked by their likelihood of being APT-related and presented to operators for manual inspection (Step 8, Figure 18). This allows network operators to prioritize their limited time on the most relevant threats.

6.3 Methodology

In this section we discuss the internals of `ghost&rae`. First, we explain the type of data we collect. We then move into the two learning modules, where we discuss our modeling and clustering processes. Finally, we conclude by discussing how the user of the system can tune and make use of `ghost&rae`'s daily reporting.

6.3.1 Datasets

In addition to the datasets described in Section 2.1.2, `ghost&rae` also requires: known APT infrastructure and an autonomous system to IP address mapping. The first is used to bootstrap our ground truth for building `ghost&rae`'s modeling component. The latter is used to extract the features, and to build the relations for the unsupervised component.

The reports described in Section 2.1 are used to build our initial ground truth. In industry reports, *indicators of compromise* (IOC) are typically provided in the form of domain names and/or IP addresses known to support APT infrastructure. We model known APT infrastructure behavior over time to detect and cluster future threats. When our system identified additional APT infrastructure, we added these to our ground truth when we could find third-party reports confirming them as IOCs for APT threats.

Every IP address belongs to an *autonomous system* (AS), which advertises routes, a collection of *BGP prefixes*, which inform other ASes how to route packets to the IP addresses contained within the advertising AS. In a nutshell, this is how packets are routed to a given IP address. These rough collections of IP addresses form logical groupings. By parsing public data from Route Views [110], we maintain a database that, given an IP address, can output the AS, BGP prefix, AS name, and the country code the IP address is allocated to.

6.3.2 Supervised Learning

The supervised learning component in `ghost&rae` is composed of two modules: the feature extraction and APT modeling modules. These modules work together to produce two reports. The first is a per-zone APT report, which helps the operator of the system determine the per-zone probability of being APT-related. The next report is a per-resource record report, where the system will provide an APT confidence score for all the classified RRs.

6.3.2.1 Feature Extraction Module

We focus on a class of APT threats that are: 1) manually operated, 2) primarily DNS-based, and 3) make use of compromised machines as "stepping stones." With these assumptions and the intuitions we draw from publicly disclosed targeted attacks, we identify several families of statistical features that can be used to model DNS behavior indicative of this class of APT threats. These features are summarized in Table 15. Potential consequences of evading the described features are discussed in detail in Section 7.1.2.

Most features are computed for a given domain name d, a resource record (RR) that is a tuple (d, ip) of domain d and its resolved IP address ip, $d \rightarrow_r ip$, or the set of IP addresses IP that d has ever resolved to $d \rightarrow_r IP$. A visual representation of infrastructure supporting both APT and non-APT domain names is shown in Figure 19.

From the beginning of our feature selection process we considered the idea of feature robustness. Specifically, we choose features that could only be evaded with a non-negligible "cost" to the adversaries, such as increases in noise, monetary and

Table 15: Summary of APT modeling features

Family	Description	Formal Description
On/Off	Distribution of lengths of successful/unsuccessful resolutions.	$\overline{C_s}, median(C_s), var(C_s)$, $\overline{C_u}, median(C_u), var(C_u)$
Infrastructure Distribution	For given d, network distribution of $d \rightarrow_r IP$.	$ASN(d)$, $BGP(d)$, $ASNAME(d)$, and $CC(d)$
Client Lookup Distribution	For a given d, the distribution of resolutions by corporate or enterprise (ent)/ISP (isp) recursive DNS sensors, the proportion of enterprise to ISP, and the count.	$\|S_{ent} \cup S_{isp}\|$, $\frac{\|S_{ent}\|}{\|S_{ent} \cup S_{isp}\|}$, $\overline{RP_{isp}}, median(RP_{isp}), var(RP_{isp})$, $\overline{RP_{ent}}, median(RP_{ent}), var(RP_{ent})$
DNS Zone Features	# resolutions, DynDNS domain, non-routable resolutions, e2ld domain zone count	-

managerial costs, raising the detection and attribution profile, and causing a decrease in the overall network agility of the infrastructure. Thus, the attackers must choose between either “manual-and-slow” functionality or open themselves to detection by existing techniques (e.g., reputation systems). We briefly explain the difficulty attackers face when attempting to evade the features after describing each feature below. A summary of the consequences attackers face from possible evasions as well as the importance measured by the random forest classification algorithm [20] is shown in Table 22.

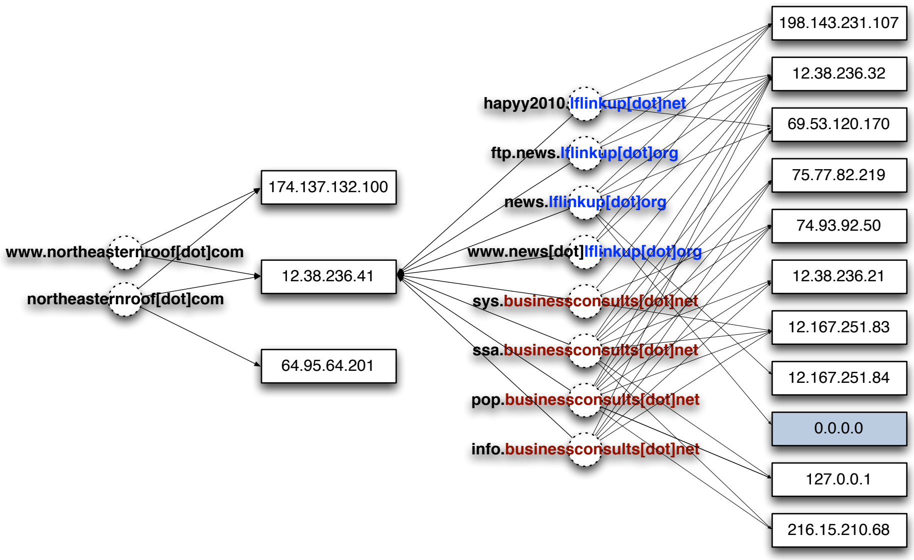

www.northeasternroof[dot]com
northeasternroof[dot]com
174.137.132.100
12.38.236.41
64.95.64.201
hapyy2010.lflinkup[dot]net
ftp.news.lflinkup[dot]org
news.lflinkup[dot]org
www.news[dot]lflinkup[dot]org
sys.businessconsults[dot]net
ssa.businessconsults[dot]net
pop.businessconsults[dot]net
info.businessconsults[dot]net
198.143.231.107
12.38.236.32
69.53.120.170
75.77.82.219
74.93.92.50
12.38.236.21
12.167.251.83
12.167.251.84
0.0.0.0
127.0.0.1
216.15.210.68

Figure 19: Infrastructure that supports both non-APT (left) and APT (right) domain names. Note the use of localhost IP addresses (127.0.0.1), multiple child labels, and high infrastructure distribution.

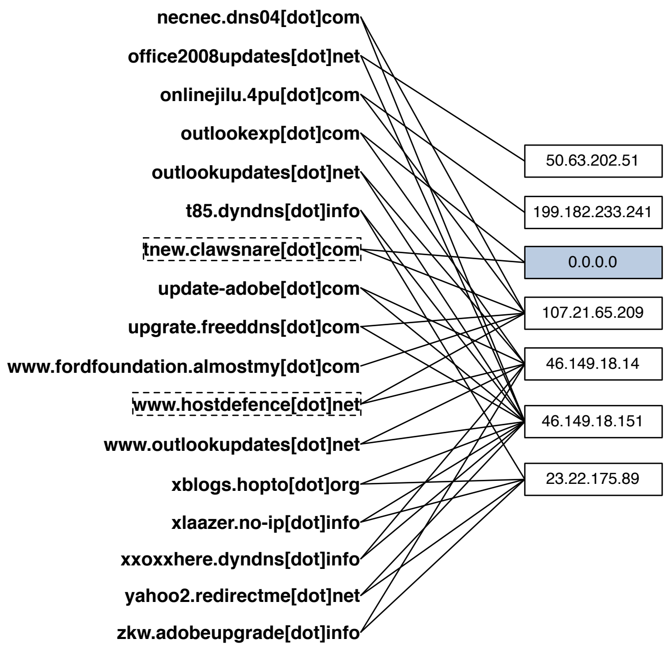


Figure 20: Example cluster showing infrastructure relationship between known APT domains (dotted line border) and unknown domains.

On/Off Behavior In order to "lie low" and prevent detection, APT threats commonly disable domain names by having them resolve to local and/or non-routable IP addresses (e.g., `127.0.0.1`) when the attackers are not actively using them. Consider an APT attacker that makes use of a common off-the-shelf remote access tool (RAT) malware. Most tools periodically "phone home" to their command and control (C&C) server, generating network traffic that may trigger an alarm. If the domain name used for the C&C, however, resolves to a non-routable IP address, this traffic never leaves the perimeter network and evades detection at the network's edge. When the attacker wishes to examine the network or exfiltrate sensitive data, she manually changes the domain to resolve to one of many compromised hosts, allows the malicious activity to complete, and "turns off" the domain name to resolve back to the "benign" host. Intuitively, we expect this to differ by being "more sporadic" than typical malicious or popular domains, that will resolve regularly. APT actors that attempt to evade this feature family will have malware that is far noisier than they currently are, putting them in-line with the noise generated by most malware threats. This may also open them up to simpler detection systems or ones looking for periodicity (see Table 22).

Given sufficient pDNS visibility, we can identify these gaps and contiguous resolution patterns and use them as features to detect APT behavior. For a given RR on day t, we consider a resolution *successful* if it was resolved on t by at least one host in our network to a routable IP address and *unsuccessful* if we did not see it resolve. We can model the on/off behavior by using the distribution of contiguous chunks of successful and unsuccessful resolutions as features, characterized by their mean, median and variance in lengths. More formally, consider the bitstring $R_{d,ip} \in \{0,1\}^n$ that denotes n days of resolution behavior of domain d to IP address ip, where 1 and 0 bits denote a successful or unsuccessful resolution, respectively. Let C_s and C_u be lists containing the lengths of these runs for successful and unsuccessful resolutions, respectively. For example, for $R_{d,ip} = 100011001$, we have $C_s = [1,2,1]$ and $C_u = [3,2]$. For $R_{d,ip}$, the computed features are $\overline{C_s}$, $median(C_s)$, $var(C_s)$, $\overline{C_u}$, $median(C_u)$, $var(C_u)$.

Infrastructure Distribution Intuitively, we expect an APT domain name to resolve to a wider variety and more distributed hosts than a non-APT domain name due to the reliance on compromised infrastructure. This follows for two reasons: attackers are unable to choose the location of hosts that are vulnerable to compromise, and they must maintain an excess of compromised hosts in case some are remediated by their rightful owners. This results in APT domain names resolving to more, geographically disparate IP addresses than those that belong to non-APT domains. Consider a domain name for a small business: it will likely have one IP address for its website, and possibly another for email hosting, but rarely many more. Attackers could evade this feature family by using bulletproof hosting in lieu of compromised machines as more commodity threats do, however, this increases the likelihood of attribution through money trail. Plausible deniability is very important for APT threats due to their potential geopolitical consequences (see Table 22).

For a given domain name, d, and the set of IPs it resolves to $d \rightarrow_r IP$, we calculate the distinct number of autonomous system (AS) numbers, BGP prefixes, AS names, and country codes the IPs in the set IP belong to. Formally, we represent these as $ASN(d)$, $BGP(d)$, $ASNAME(d)$, and $CC(d)$, respectively. We extract this information from the local BGP database we built using the Route Views archive [110]. Similar data can be collected from Team Cymru's [173] IP to ASN mapping service. Overall, we expect these features to be higher for APT domains than non-APT domains.

Client Lookup Distribution Intuitively, we expect APT threats to be "targeted" with respect to the clients that query and use its infrastructure. Recall that our passive DNS data contains which enterprises' and ISPs' DNS resolvers resolved a given resource record. APT domains should be queried disproportionately more by enterprise DNS recursive resolvers, and occasionally for short periods of time from ISP DNS recursive resolvers due to work devices being brought home. For non-APT domains, we expect popular benign domains and traditional malicious domains to be queried from everywhere with high regularity, and unpopular benign domains to be queried infrequently, but primarily from ISP sensors. Attackers could "poison" this feature by generating spurious lookups from around the globe, but this will raise the noise and operational profile of the threat significantly and open the threat to more traditional detection systems (see Table 22). With more eyes on the threat, it stands to reason it will be more likely to be detected elsewhere.

For a given RR (d, ip), our passive DNS data contains the set of enterprise DNS recursive resolvers, S_{ent}, and the set of ISP DNS recursive resolvers, S_{isp}, that resolved

(d, ip). Furthermore, $\forall \int \in S_{ent} \cup S_{isp}$, we know the first date and last date $d \rightarrow_r ip$ by a DNS recursive resolver $\int$, which allows us to compute the *resolution period*, $\Delta_{\int,(d,ip)}$. The first two features in this family are the unique number of DNS recursive resolvers, $|S_{ent} \cup S_{isp}|$, and the proportion of enterprise to ISP DNS recursive resolvers, $\frac{|S_{ent}|}{|S_{ent} \cup S_{isp}|}$, that resolved (d, ip). The subsequent features capture the distribution of the resolution period for an RR over enterprise and ISP DNS recursive resolvers. Let RP_{ent} be a vector containing $\forall_{\int \in S_{ent}} \Delta_{\int,(d,ip)}$ and an analogous vector RP_{isp} for ISP DNS recursive resolvers. As we did for the On/Off feature family, we compute $\overline{RP_{isp}}$, $median(RP_{isp})$, $variant(RP_{isp})$ and $\overline{RP_{ent}}$, $median(RP_{ent})$, $var(RP_{ent})$ as features. We expect APT domains to be resolved more commonly and for longer periods of time by enterprise sensors than ISP sensors.

DNS Zone Features In isolation, these features are generally weaker than the previously described features, but when paired with the earlier features they improve the accuracy of `ghost&rae`'s modeling. The remaining features are: the number of times (d, ip) successfully resolved, whether d uses a dynamic DNS (DynDNS) provider, the number of non-routable IP addresses in IP where $d \rightarrow_r IP$, and the number of domains under d's effective second-level domain zone (E2LD). Evading these features is possible, but increases the monetary cost to attackers when setting up their infrastructure eschewing DynDNS (free) domains for paid ones and having to pay for additional E2LDs rather than using child labels under one paid-for E2LD zone. Furthermore, by paying for more assets it increases the attribution footprint for the attack, which further strain international relations. Evading the others increases the noise of the attack by having more total resolutions (global noise increase) or fewer resolutions to bogons (network-specific noise increase).

The number of successful resolutions of (d, ip) captures the difference between popular domains and commodity C&C domains, which will resolve many times, and unpopular and APT domains, which resolve relatively fewer times. DynDNS provides free subdomains, which are used commonly by traditional malicious and APT threats, and less frequently by unpopular and popular domains. The number of non-routable IPs resolved by d helps differentiate between a domain name that is sporadically resolved from one that is disabled by resolving to a non-routable IP address. Understanding the effective second-level domain zone feature requires a brief explanation of the DNS system.

Let $Z(d)$ be a function that returns the *effective second-level domain* or (e2ld) of a fully qualified domain name d. $Z(d)$ represents the shortest ancestor of d that is operated by a single party that is not a registrar or DynDNS provider. For example, consider $d =$ `random.google.com`. $Z(d) =$ `google.com` because Google, Inc. owns `google.com`, which it leases from the owner of the `.com` top-level domain (TLD). Now, let $d =$ `more.random.dyndns.org`; in this case, $Z(d) =$ `random.dyndns.org` because `dyndns.org` is a Dynamic DNS (DynDNS) provider and offers subdomains under `*.dyndns.org` (or any other e2LD zone under the same authority) for free. Not only are DynDNS domains commonly free, they also help obfuscate the owner's origin since there is no attribution (i.e., WHOIS) information for free DynDNS domains. The e2ld domain zone count feature is the number of subdomains where $Z(d)$ is a parent domain, which helps separate benign domains that typically have few child domains (e.g., `www.`, `mail.`, etc.) from APT domains which heavily use subdomains.

6.3.2.2 APT Modeling Module

After we extract the above features using the Feature Extraction Module, we train three models to classify infrastructure as being APT-related or not: Naive Bayes, generalized Linear Regression, and Random Forest. We use these models to classify RRs and create our two APT reports. We evaluate the models in two ways. First, we perform 10-fold cross-validation on each model both to show that our features are meaningful and that they can be modeled successfully. Next, we experimentally identify the *time-to-detection* (TTD) of APT infrastructure.

We evaluate the APT Modeling Module by the machine learning community standard of k-fold cross-validation (CV) [53] based on the known labels from our APT Knowledge Base, and also use the k-fold CV results to empirically select the best model. k-fold cross-validation is an evaluation technique that partitions the data into k equal chunks, trains a model against $k-1$ chunks, tests against 1 chunk, and averages the accuracy of the model when training/testing using each arrangement of chunks. We can compare the results of k-fold CV to determine our best model to use. For our experiments, we choose $k = 10$, which is standard in the machine learning community.

To determine the TTD of the APT Modeling Module, we reserve an early segment of our ground truth for training, and model and test against future resource records as we increase our visibility to determine at what "age" we can detect APT domains in general. As explained in Section 6.3.1, our passive DNS observation period extends as far back as January 1st, 2011 until the present day. Our ground truth resource records cover this entire period. We reserve resource records observed during 2011 and 2012 for training, and test against successive cumulative months in 2013, i.e., we test against January, 2013; January and February, 2013, and so on until we finally test against all of 2013. In other words, how long until we can detect APT threats within a network with high accuracy and few false positives. By plotting the average accuracy, true positive, and false positive rates for successive months, we can identify the age with which we can classify APT domains with reasonable accuracy.

6.3.2.3 APT Reports

The APT Modeling Module outputs two kinds of APT Reports: an RR-based Report and a Zone-based Report.

The RR-based report is a collection of resource records (RR), a predicted classification label (`apt` or `nonapt`), and the model's predicted likelihood that the RR is APT-related ($[0 - 1]$, where 0 is certainly `nonapt` and 1 is certainly `apt`). This report is primarily intended for input into the Unsupervised Learning component of `ghost&rae`, and an aggregated version of the RR-based Report is generated for human consumption. This aggregation is presented in the Zone-based Report.

The Zone-based Report is a collection of e2ld zones, a predicted classification label, and the model's average predicted likelihood over the RRs seen under the e2ld zone. The RR-based Report will commonly output many RRs, which makes an aggregation by zones more useful. Consider the example in Figure 19. While there are many unique RRs for the APT infrastructure on the right side of the figure, the entire infrastructure can be represented using only three e2ld zones (`businessconsults.net`, `lflinkup.net`, and `lflinkup.org`). If a majority of RRs under an e2ld are classified as `apt`, the zone is classified as `apt`, `nonapt` otherwise.

6.3.3 Unsupervised Learning

In the unsupervised component of `ghost&rae` we make use of the APT-likelihood scores to build "communities" of APT domain names that are similar in two ways: 1) the domain names pointed to the same infrastructure over a period of time, and 2) the average APT-confidence of domain names in these communities should be high. In other words, the clustering methods should be able to provide us with better correlations between domain names under the same targeted attack or APT group and consist primarily of highly likely APT domain names outside of our ground truth.

To achieve this we need to be able to cluster RRs from the entire IPv4 space. To process this much data, we have to transform and cluster the data we receive from the RR-based APT report, and properly rank the produced domain name clusters according to their relevance to our APT models. It is worth noting that it is a bad idea to only cluster RRs that have high confidence to the known APT models we have built using the publicly available ground truth. If we do this, we may exclude new APT domain names that use known APT infrastructure we have in our ground truth, but which do not have enough passive DNS history to be properly classified by our APT models. Thus, we need to include all of the RRs in our RR-based APT reports.

6.3.3.1 Spectral Clustering Module

To cluster related APT infrastructure, we perform steps five through seven in Figure 18. The seed for the clustering process is the RR-based APT report. Using the domain names, IP addresses and the probability for the RR to be APT or not we build a weighted incidence matrix relating domain names to IP addresses. Next, we will factorize the matrix for efficiency reasons, and cluster the resulting factorized matrix. Before we explain how we achieve all these, we provide the intuition behind this process with a real APT example.

Consider a bipartite graph with IP addresses on one side and domain names that resolve to them on the other, as in Figure 20. As a general construction, one expects there to be "natural" clusters of domain names and IP addresses that are related in some way. The example in Figure 20 shows one such cluster where the domain names with the dotted line border are known to be affiliated with a particular APT operator, while the domains without a border are of unknown use. By clustering these relations, we can group similar domain names by the infrastructure they use. Based on our assumptions we expect APT and non-APT domains to share infrastructure, so we must provide additional information to a clustering scheme to separate them. We use the RR-based APT output to inform our clustering process to accomplish this separation.

The input to the spectral clustering module is the RR-based APT dataset $rrAPT^{\tau}$

$$M^{\tau} = \begin{array}{c c} & \begin{array}{ccc} ip_1 & \cdots & ip_k \end{array} \\ \begin{array}{c} d_1 \\ d_2 \\ \vdots \\ d_n \end{array} & \begin{pmatrix} w_{1,1} & \cdots & w_{1,k} \\ w_{2,1} & \cdots & w_{2,k} \\ \vdots & \ddots & \vdots \\ w_{n,1} & \cdots & w_{n,k} \end{pmatrix} \end{array}$$

Figure 21: The M^{τ} $n \times k$ sparse matrix that will be used in `ghost&rae`'s SVD process during temporal window τ for n domain names and k IPs.

from period τ. More formally we have, $rrAPT^{\tau} = \{d_i, ip_j, P_{apt}(d_i, ip_j)\}_{i=1..n, j=1...k}$, where d is the domain name in the RR, ip is that IP address in the RR and $P_{apt}(d, ip)$ is the APT model's confidence that the RR is APT-related. From the same RR dataset $rrAPT^{\tau}$, we can derive the sets RRD^{τ} and $RRIP^{\tau}$. The set RRD^{τ} reflects the set of domain names in $rrAPT^{\tau}$, where $rrAPT^{\tau}$ is the set of IPs in $rrAPT^{\tau}$. This confidence is bound between 0 and 1. 0 means the model is certain the RR is not APT-related and 1 means the model is certain the RR is APT-related. We use this to weight the incidence matrix that corresponds to the bipartite graph we previously discussed.

A bipartite graph between the RRs found in the $rrAPT^{\tau}$ dataset can be represented as a sparse adjacency matrix M^{τ}. In this matrix (see Figure 21), the rows and columns correspond to the left and right vertices and a non-zero value at j, i of the matrix M^{τ} corresponds to an edge between the j^{th} left vertex and the i^{th} right vertex. We define as $w_{i,j} = (2P_{apt}(d_i, ip_i) - 1)^2$, if RR $(d_i, ip_j) \in rrAPT^{\tau}$. If the RR is not in the $rrAPT^{\tau}$ dataset, then the $w_{i,j} = 0$. We define the weight $w_{i,j}$ in this way to penalize RRs that have a lower confidence of being APT.

In general, this matrix will be very large, making it difficult to process. Since the matrix will be very sparse, we can reduce the matrix using *singular value decomposition* (SVD). This operation will change the computational complexity from $\mathrm{O}\big(|RRD^{\tau}| \times |RRIP^{\tau}|\big)$ to $\mathrm{O}\big(|RRD^{\tau}| \times \varrho\big)$, where ϱ is the number of the first eigenvalues we will get from factorizing the matrix M^{τ}. Using SVD enables us to efficiently

```
INPUT   : Sparse matrix $M \in \Re^{n \times k}$, as can be seen in Figure 21.
[1] : Normalize $M$: $\forall j = 1,..,k \quad \sum_{i=1}^{n} M_{i,j} = 1$
[2] : Compute the similarity matrix $S$ from $M$: $S = M^T \cdot M$
[3] : Compute the first $\varrho$ eigenvectors from $S$ by eigen-decomposition.
Let $U \in \Re^{\varrho \times n}$ be the matrix containing $n$ vectors $u_1, ..., u_n$ of size $\varrho$ resulting from the
eigen-decomposition of $S$ (a vector $u_i$ is a reduced $\varrho$-dimensional representation of the $i$-th
domain name).
[4] : Cluster the produced domain name vectors $\{u_i\}_{i=1,..,n}$ using the X-means
algorithm [127], with a minimum number of clusters = 1 and a maximum number of clusters
= 1,000.
[5] : Sort clusters by Average APT Probability from the domain names in each produced
cluster.
[6] : Exclude already labeled clusters.
OUTPUT: Clusters of likely APT domains, sorted by average model confidence.
```

Algorithm 3: An algorithmic representation of the unsupervised component in `ghost&rae`. Spectral clustering and cluster characterization processes in `ghost&rae`, over a temporal period τ and using the domain name to IP matrix M^τ.

model the RR-based dataset. The only variable in our clustering process is the size of the $|RRD^\tau|$ set, which tends to be stable over short periods of times (i.e., months).

6.3.3.2 Cluster Characterization Module

Since we are likely to generate many clusters it would be infeasible to manually inspect all of them if we wish to yield tangible and useful results. In order to prioritize the efforts of operators, before we present the clusters to the operators of `ghost&rae` we sort them in descending order according to the average APT-probability score from the domain names in the cluster. Next, we exclude all clusters that only contain domain names that are already labeled. Thus, the operator, after the unsupervised learning component has finished, will receive a ranked list APT clusters (Step 7, Figure 18) for review. These reports are meant to be manually inspected by network operators to further improve the APT Knowledge Base and constantly update our ability to model APT infrastructure. Operators of `ghost&rae` can tune the number of clusters presented to to cater to their organization's size. This is to prevent overwhelming smaller organizations, or to allow for in-depth investigation by dedicated security response teams if present.

Table 16: 10-fold cross validation runs for different models in the APT Modeling Module (averaged across 10 runs).

Model	**Acc. (%)**	**TPR (%)**	**FPR (%)**
Naive bayes	70.18	92.30	41.79
Linear regression	98.44	91.23	1.11
Random forest	99.84	97.92	0.04

6.4 Evaluation

In this section, we evaluate `ghost&rae`. For the supervised phase, we perform model selection to identify the best model to use in practice. Furthermore, we present the results of the time-to-detection (TTD) experiment that empirically derives the length of time it takes to begin successfully detecting APT infrastructure. For the unsupervised phase, we explain the results of the clustering output by using `ghost&rae` to model/cluster all of IPv4. Specifically, we discuss the improvements made to our ground truth using its reports, and the structure and size of the clusters in relation to their average APT score. From these results, we discuss some interesting APT case studies, including one previously undiscovered threat in Section 6.5.

6.4.1 Supervised Phase Evaluation

We perform 10-fold cross-validation on our ground truth dataset using three modeling algorithms: random forests, generalized linear regression, and naive bayes. 10-fold cross-validation is a standard measure for a model's accuracy and is described in detail in Section 6.3.2.2. We present the results by showing the average accuracy, true positive rate, and false positive rate. A table of these comparisons are presented in Table 16 and a ROC-curve comparing the performance of the models is shown in Figure 22. Our ground truth contains 47,952 resource records, of which 24,531 are APT-related and 23,421 are non-APT-related. These RRs fall under 11,549 e2ld zones as described Section 6.3.2.1.

Random forest outperforms all the other models as clearly shown by Table 16 and

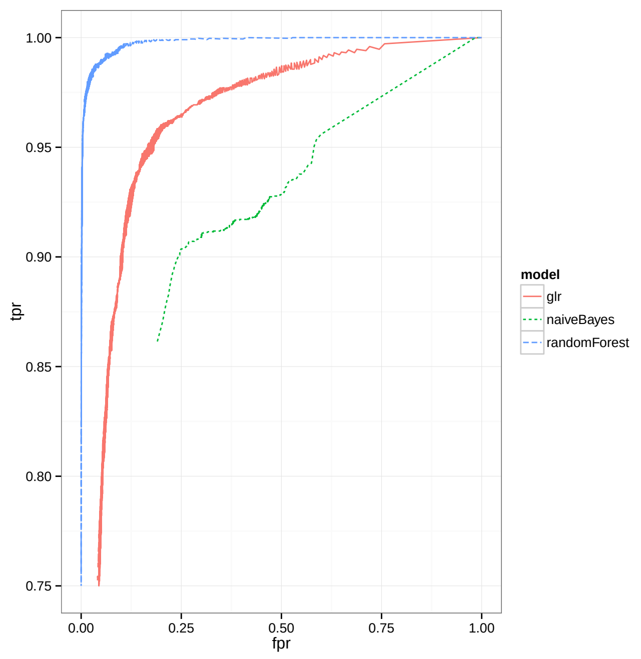


Figure 22: ROC-curve of different models for the APT Modeling Module.

Figure 22. In general, true positive rates are high for all models, but false positive rates are poor for glr and exceedingly poor for naive bayes. The exceedingly poor performance of naive bayes was initially disconcerting, but makes sense given the independence assumption made by naive bayes. Feature families for APT detection are not independent. Consider the feature family for the client lookup distribution (see Table 15). A clean split exists only between popular/traditional malicious domains and APT-related/unpopular domains, requiring other dependent features to help further differentiate APT domains from unpopular domains. This does not preclude the use of these features, especially since they reduce false positive rates, but it does necessitate using a model that does not assume feature independence.

Since many of our features rely on historic resolution evidence, it is important to know when our models can accurately detect APT behavior in an RR after it has started resolving in our networks. We determine the time-to-detection (TTD) for the modeling and features we use in `ghost&rae`. We anticipate that random forest will continue to outperform the other models, but we include the others for completeness.

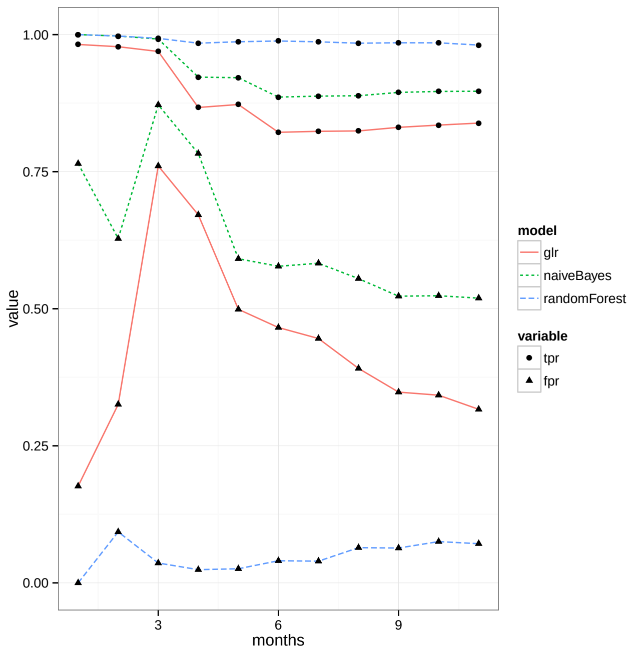


Figure 23: Time-to-Detection for APT-related domains (TP/FP rates) for our chosen models.

Figure 23 shows the results of this experiment. The true positive rates (TPR) shown in the top of Figure 23 show largely consistent TPRs for the random forest model, and a dip in performance for the other models. We initially expected the TPR to increase over time, but our ground truth contains disproportionately more domains starting to be used around March, 2013 resulting in the drops in TPR around this time. The earlier domains were mostly self-contained within the first few months so were more accurately modeled without future observations. Since TPR are largely stable for the random forest, we make our time lag decision based on the FPR. Once again, we see false positive rates (FPR) start off strong while there are fewer APT-related domains coming "online." The lowest FPR for the random forest model occurs during April. For our IPv4 experiment, we will only consider domains with at least a four month observation period to focus on resource records we are mostly likely to classify accurately. The fact that naive bayes and glr perform substantially worse with respect to FPR further motivates our choice to use random forest in `ghost&rae`.

Table 17: Count of AV labels of malware that communicate with domains in the Top 50 clusters.

AV Label	Count
Generic Trojan	1,691
Generic Backdoor	463
Generic Downloader	95
Poison Ivy	60
Tufik	49
DarkSnow	47

To summarize, `ghost&rae` has high true positive detections nearly immediately, but if an organization can afford to wait four months to let RRs age the classification results will contain fewer false positives.

Malware Related to High Confidence APT Clusters In general, APT threats only need a remote access tool (RAT) to achieve their malicious ends. APTs are primarily focused on extracting information, and RATs allow this to happen remotely. As such, we commonly see APT threats making use of common off-the-shelf malware. For the top 50 clusters generated by `ghost&rae` that began in November 2013, we present the Anti-Virus labels of malware known to query these domain names in Table 17. If the labels largely indicate RATs and trojans, it gives further evidence that these are in fact APT-related beyond the APT Modeling Module confidence scores. However, if we see a predominance of non-generic malware families (Zeus, Bobax, ZeroAccess, Virut, etc.) the infrastructure likely represents commodity malicious infrastructure.

We see a predominance of generic classifications for the AV labels that contacted domains in our likely APT clusters, the bulk of which are trojans or backdoors. Since APT operators are known to use such common tools, sometimes with slight modifications, these are the AV labels one would expect to be assigned to malware used in APT threats. Poison Ivy occurs relatively frequently and is a popular RAT kit used by APT operators [104]. The final two families are generic AV detections for families of network-spreading malware. While this makes sense as well from the APT standpoint, we are investigating these clusters further to see the relationship and use of these families in APT threats. The remaining labels are frequently heuristic detections and other generic labels.

6.4.1.1 Manual Labeling Comparison

We can compare the gains provided by `ghost&rae`, compared to manual labeling, in two ways: time and accuracy. In the context of time, we can estimate how long it would have likely taken to label all the domains that were detected by `ghost&rae`. With respect to accuracy, we can identify the differences between false negatives and false positives of the two detection "systems."

Industry reports suggest that, given the number of alerts an organization receives, security analysts have seven minutes to investigate the validity of an alert [8]. If these events were given back-to-back to a security organization without any other alerts in between, it would take roughly 52 days to achieve the same ground truth gains `ghost&rae` was able to do in fewer than five hours.

As far as accuracy is concerned, `ghost&rae` increased the ground truth coverage by 408% from the manually labeled entries. This means manual labeling incurred an additional 10,716 false negatives compared to `ghost&rae`. This is to be expected as an automated system can cover more ground than human analysts. However, this naturally comes with an increased likelihood of false positives. Using our best model during evaluation, `ghost&rae` had 462 additional false positives compared to zero from the manual labeling process. If the above investigation statistic still holds, this could cost an organization 2.25 days of investigation but at the gain of many more missed detections and days of analysts' time saved.

6.4.2 Unsupervised Phase Evaluation

The unsupervised phase of `ghost&rae` is focused on providing actionable clusters to network operators to expand their understanding of APT threats and iteratively improve the APT Knowledge Base. In turn, this increases the overall accuracy of the system. We model *all* of IPv4 using `ghost&rae`. To illustrate its effectiveness, we explain how the system was used to improve the size of our APT Knowledge Base while in operational use. We show that network operators confirm it successfully detects APT campaigns.

In total, we classified and clustered approximately 302 million resource records, yielding roughly 1.1 million clusters. A distribution of the average APT score of the clusters is shown in Figure 24. While we have many clusters, a tiny proportion have high chances of being APT related and should be investigated. Of the original 1.1 million clusters, 37,164 have a score ≥ 0.5, 724 have a score ≥ 0.8 and only 60 have a score ≥ 0.9. Manual analysis is time consuming, but there are very few high likelihood clusters operators must investigate *even* when modeling the entire internet. Examining the relationship between cluster sizes and their average APT confidence shows that high confidence clusters tend to be small and thus easier to manually verify (see Figure 25). We present some example clusters in Section 6.5. In general, the reader will observe that they are small, the domains are highly related, and there is rarely confusion between the operators if the clusters are at least partially known internally.

We ran `ghost&rae` through the entire process illustrated in Figure 18 for a period of three month. Using the final output from Figure 18 (7), we manually inspected the generated clusters in the order described in Section 6.3.3.2 to iteratively improve our understanding of APT threats. Our initial APT Knowledge Base consisted of only 2,831 domain name zones known to facilitate APT infrastructure. In only three month, we were able to improve our APT Knowledge Base by 408% to arrive at

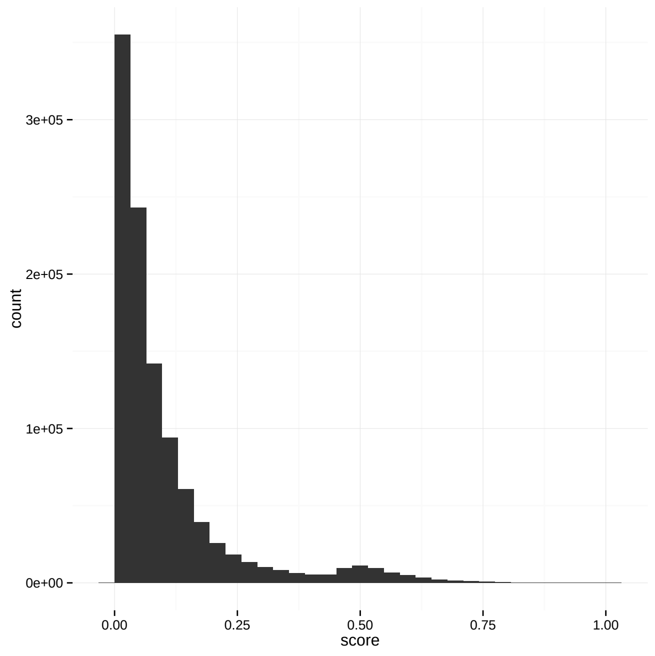


Figure 24: Distribution of average APT confidences from unsupervised evaluation of all IPv4. Most clusters are highly unlikely to be APT.

the 11,549 known labeled APT zones through confirmation by third party industry reports or by drawing conclusive relationships between the infrastructure, i.e., the APT threats shared E2LD domain names, hosting infrastructure with known APTs, and/or had identical WHOIS information. In addition to increasing our ground truth coverage of APT threats using the IPv4 experiment, we also identify additional infrastructure for known threats, threats that have continued to function in spite of previous reports, and discover a previously unknown APT threat. We detail these case studies in Section 6.5.

6.5 Case Studies

In this section, we discuss three case studies of APT infrastructure `ghost&rae` identified during our IPv4 experiment. In many cases, we found additional infrastructure for known APT threats, or show that previously thought inactive APT threats continue to expand and use their infrastructure. In one case, we identify what is to

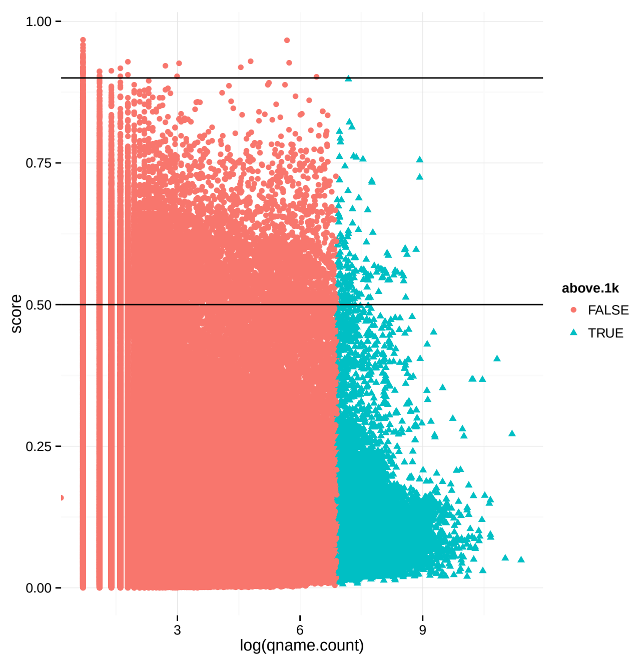


Figure 25: Number of qnames in a cluster (log-scale) vs. average APT confidence for all of IPv4.

the best of our knowledge one new APT threat that originates in China and targets nearby East Asian countries.

6.5.1 Mutter/Beebus

Our first case study confirms an earlier finding by FireEye when they reverse engineered the Mutter backdoor and using the C&C infrastructure identified it as being related to Operation Beebus [78, 179]. We independently identified this APT infrastructure (see Table 18) without reverse-engineering malware and found that despite being found over a year ago, the C&C infrastructure is still active as of November 10, 2014 and resolves to 114.52.1.21 an IP address in South Korea's SK Telecom. The link between our identified infrastructure and that discovered by FireEye was the email used to register the domains: `binalakshminp@yahoo.com`. Registration emails used in WHOIS tend to be strong indicators of relatedness in the event domain's WHOIS is not privacy protected. There are an additional 51 domain names under

Table 18: Mutter/Beebus Cluster

Domain Name
adobe.dsmtp[dot]com
adobe.winsupdate[dot]com

Table 19: Indian Research Lab Cluster

Domain Name
gov.erpds[dot]com
hqr.erpds[dot]com
updata.erpds[dot]com
report.erpds[dot]com
support.erpds[dot]com

the dsmtp[dot]com zone that resolved through 2014, but are no longer active.

6.5.2 Sinon Campaign

Our second case study used some of the same infrastructure as Operation Beebus, but instead targeted research laboratories in India, specifically, India's Defense Metallurgical Research Laboratory and Defense Research and Development Organization. While `ghost&rae` discovered this attack in mid-2014, it has since been discovered and confirmed by 3rd party sources with additional information about the infection vector [34]. The attack was distributed by a spear-phishing email that took advantage of CVE-2012-0158. The domains we identified are shown in Table 19. Once again the domains include innocuous child labels, e.g., `gov` and `support`, in an attempt to not raise any red flags given the network being targeted.

6.5.3 Domain Name Family Campaigns

Our third case study identifies additional, previously undisclosed infrastructure explained in the extended clustering report [67] of Hardy et. al.'s recent work in characterizing politically motivated APT threats [68]. These campaigns targeted four groups, one in China and three in Tibet, focused on human rights issues in China and human rights issues in Tibet, respectively. The authors state they only observed the attacks from November 2010 to March 4, 2013.

Table 20: Domain Name Family Campaigns Cluster

Domain Name
telnet.asianewsnow[dot]com
smtp.asianewsnow[dot]com
pop3.asianewsnow[dot]com
www.livemailagent[dot]org
www.asianewsnow[dot]com
www.mailruagent[dot]org
proxy.asianewsnow[dot]com
web.asianewsnow[dot]com
dns.asianewsnow[dot]com
firewall.asianewsnow[dot]com
asianewsnow[dot]com
bangju.codns[dot]com
IP Addresses
115.28.58[dot]35
223.25.241[dot]39
37.61.54[dot]158
78.16.49[dot]15
223.25.241[dot]39
8.7.198[dot]45
59.24.3[dot]173
46.82.174[dot]68

Using `ghost&rae` we were able to identify more infrastructure than reported in [68, 67], but also discovered that the attack is still active. In addition to the infrastructure shown in Table 20 there are 25 additional domain names that are actively resolving as of July 15, 2015 to IPs in China, Malaysia, Azerbaijan, the United States, South Korea, and Germany.

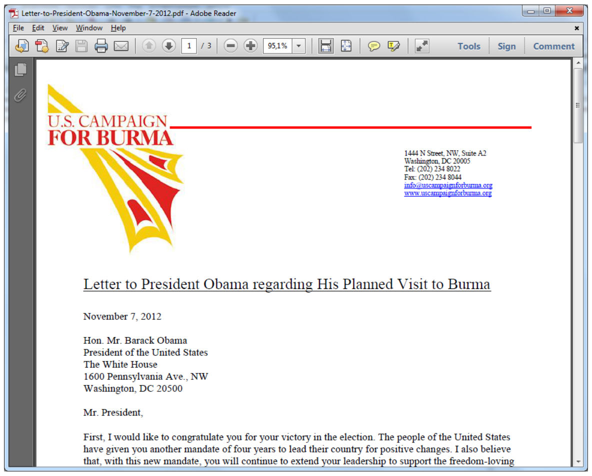
6.5.4 New APT Threat—Espionage

Our final example is to the best of our knowledge a new APT campaign or campaigns that has not been fully discovered before. While we have no conclusive evidence, much of the circumstantial evidence points to it being one or more politically motivated APT threats targeting countries in East Asia and likely perpetrated by China. The domain names contained within the cluster are shown in Table 21. All but one (msgkmg.com) of the domains appear to be affiliated with the campaigns — as a benign website that shares hosting with APT campaigns.

The three main indicators are the domain names themselves, malware evidence found in relation to these domain names, and baiting techniques that suggest political motivations. First, nearly all of the domain names use popular Chinese Dynamic DNS providers and use third level labels for organization or deception purposes (see bold in Table 21). Specifically, domain names like `office-update.eicp[dot]net` are meant to look innocuous as though they are updates to Microsoft Word and domains such as `thudohanoi.gnway[dot]net`, `timesofindia.oicp[dot]net`, and `yangontrip[dot]com` are suggestions on the intended target. In all, the domain names "call out" the countries, or cities in the countries, of India, Vietnam, Myanmar, the Philippines. Also alluded to in the domain names are a Vietnamese university, news agency, and shopping network. 58 of the domain names still actively resolve as of November 11, 2014 and are located on IP addresses in South Korea (17), Hong Kong (11), Mainland China (10), the United States (9), Mexico (7) Taiwan (1), Tunisia (1), Germany (1), and Belgium (1).

Second, there are numerous reports of generic trojans communicating with this infrastructure, which one would expect in an APT attack centered around reconnaissance. Numerous [44, 149, 93, 79, 159, 160, 156, 158, 157] industry reports link portions of the infrastructure `ghost&rae` automatically identified with generic malware known to include commands such as "capturing screen, audio and webcam, logging keystrokes, managing passwords, and acting as a relay server" [79]. One industry report [44] even suggested it was potentially an APT threat, but did not provide sufficient evidence to support this claim. Given the multiple malware families shown to communicate with this infrastructure, it is possible this infrastructure supports multiple campaigns.

Third, one industry report [93] identified a PDF used to initiate the infection titled "Letter to President Obama regarding His Planned Visit to Burma" (see Figure 26). The document is purported to be "from Aung Ding, Executive Director of U.S. Campaign for Burma and dated by November 7, 2012." While the authors of


Letter-to-President-Obama-November-7-2012.pdf - Adobe Reader
File Edit View Window Help
1 / 3
95,1%
Tools Sign Comment
U.S. CAMPAIGN FOR BURMA

1444 N Street, NW, Suite A2
Washington, DC 20005
Tel: (202) 234 8022
Fax: (202) 234 8044
info@uscampaignforburma.org
www.uscampaignforburma.org

Letter to President Obama regarding His Planned Visit to Burma

November 7, 2012

Hon. Mr. Barack Obama
President of the United States
The White House
1600 Pennsylvania Ave., NW
Washington, DC 20500

Mr. President,

First, I would like to congratulate you for your victory in the election. The people of the United States have given you another mandate of four years to lead their country for positive changes. I also believe that, with this new mandate, you will continue to extend your leadership to support the freedom-loving

Figure 26: Spear-phishing PDF for Espionage Threat

the report make no suggestion of this being an APT, together with the other evidence presented above we conclude `ghost&rae` has discovered a new politically-motivated APT campaign. We encourage the security community, especially those interested in politically-motivated attacks, to further investigate these domain names to better understand the motives and actions of its perpetrators.

Table 21: Likely Politically-motivated APT Cluster

Domain Name
office-update.eicp[dot]net
bkav.coyo[dot]eu
timesofindia.8866[dot]org
thudohanoi.gnway[dot]net
microsofts.eicp[dot]net
myanmartech.gnway[dot]net
dieforwhat.vicp[dot]net
aseanreg.uicp[dot]net
philipine.gnway[dot]net
myanmartech.vicp[dot]net
ahzx.eicp[dot]net
saigon.gnway[dot]net
mmkcg.uicp[dot]net
pvep.vicp[dot]net
googlemm.vicp[dot]net
myanmartrip.51vip[dot]biz
vnn.gnway[dot]net
thanglong.vicp[dot]net
vnanet.vicp[dot]net
bkav.meibu[dot]com
aseannew.8866[dot]org
kmxwt.oicp[dot]net
myanmartech.ticp[dot]net
mncgn.51vip[dot]biz
minininjia.vicp[dot]net
sayakyaw.xicp[dot]net
tcrc.gnway[dot]net
tttinhocvn.gnway[dot]net
bkavonline.xicp[dot]net
outfrozen.oicp[dot]net
vietnamnet.oicp[dot]net
office-update.oicp[dot]net
timesofindia.oicp[dot]net
ygnnptzz.xicp[dot]net
systemdata.vicp[dot]net
austcham.vicp[dot]net
greensky27.vicp[dot]net
vietnam.gnway[dot]net
yqy21.vicp[dot]net
jww998877.gnway[dot]net
hochiminh.vicp[dot]net
loveteresa.vicp[dot]net
aseedi5.oicp[dot]net
www.yahooauservice[dot]com
qiu-yong.oicp[dot]net
vietmale.vicp[dot]net
wwwdhp.gnway[dot]net
myanmarenews.vicp[dot]net
myanmarmg.vicp[dot]net
vnanet.vicp[dot]cc
ibubble.vicp[dot]net
ninjia886.vicp[dot]net
kyawthumyin.xicp[dot]net
killlab.vicp[dot]net
www.yangontrip[dot]com
yahooauservice[dot]com
yangontrip[dot]com
yqyzl.vicp[dot]net
msgkmg[dot]com

CHAPTER VII

COSTS OF EVASION

Botnet infrastructure is constantly on the move to evade detection and takedown. As techniques described in this dissertation become public, the natural question to ask is if, and how can, attackers evade such techniques? First, we discuss each component of the overall system with respect to potential avenues of evasion and the drawbacks it entails for clever attackers. Second, we discuss, in the event a botnet successfully evades a takedown initiated by our system, how much time it would take to reevaluate the new infrastructure for an additional takedown action. We show that we can likely keep up with migrating attackers due to the automation improvements the systems in this dissertation provides.

7.1 Evasion

7.1.1 Evading Infrastructure Enumeration

Malware-level Attackers are always attempting to evade newly created defenses. The most obvious ways to evade `gza` are through timing attacks, peer-to-peer validation of network resource connectivity, communicating with different protocols, or by evading dynamic analysis entirely with excessive timeouts prior to performing malicious behavior. We discuss these evasion techniques and present methods to address these shortcomings. Since our games use RFC-compliant network responses, malware is unable to determine if it is being gamed at the host-level and subsequently must use the network in clever ways to determine its execution environment.

Dynamic malware analysis systems generally execute malware for a fixed period of time, usually around five minutes per sample. Malware can remain dormant until this time passes to evade detection and analysis. Prior work addresses this limitation by finding these *trigger-based behaviors* and generating inputs to satisfy the triggers at runtime. This limitation applies to all dynamic analysis systems in general and is orthogonal to the problem we are trying to solve of increasing the network information an executing sample attempts to connect to.

Overhead incurred during usermode packet generation could enable a clever malware author to determine if they are being gamed or not. As a performance improvement, `gza` will only route packets relevant to the game in question. For example, when G_{dnsw} is being played, `iptables` will only route UDP packets with a port of 53 destined for a VM. If DNS packets take abnormally long, while packets of other types are unaffected this could alert a malware sample that it is being analyzed. Simply routing all packets through its game would apply this overhead uniformly across all packets, removing the signal.

Peer-to-peer (P2P) evasion is when a malware sample verifies the results of a DNS or TCP request by asking another infected machine to perform an identical request. If a sample, m, cannot resolve a domain name d, but fellow infected hosts can resolve d successfully, m has reason to believe it is being run under our system. Communicating this information, however, requires the network. This forces m to succumb to gameplay one way or another; gaming of its initial C&C communication or gaming of verification queries to its peers. By focusing on the building blocks of network communication, we force all network activity to be gamed.

To perform a DNS query, a malware sample could query an HTTP-based DNS tool[1], bypassing the DNS protocol entirely. Furthermore, it could directly connect to a C&C using a non-gamed protocol, such as UDP. These problems are easily addressed by running aggregate games and adding additional protocols. Querying an HTTP-based DNS lookup tool still requires *some* network activity so running DNS and TCP games *simultaneously* would prevent this lookup from succeeding. If an attacker uses

[1]`http://www.kloth.net/services/nslookup.php`

another protocol, such as UDP, it is easy to write a new game that targets this new behavior. As malware adapts to the presence of network games, malware analysts can keep pace with malware authors without too much effort.

Network-level Internet miscreants are faced with two posibilities to evade network infrastructure enumeration as described in Chapter 4: inject noise into the passive DNS or forgo the DNS entirely.

If a botnet uses the DNS to reach its command and control server, the domain name *must* resolve in order for malware to perform its malicious activities. Changing the resolved IP address for a domain name used by malware could inject noise, but at the cost of preventing *all* malware from communicating with the C&C during that period of time. In most cases, this would prevent successful monetization of the botnet and would be unlikely to be done by the malware authors. Alternatively, malware authors could spoof resolutions to C&C domains with incorrect A records, since DNS is based on UDP and is stateless and not authenticated. If the queries were performed in one of the networks where passive DNS data are collected, these spurious relationships could affect our systems. Faking resolutions to benign domains would be handled by our aggressive whitelisting (Section 4.1.2.1) or by detecting collateral damage (Chapter 6). Futhermore, attackers would need to be heavily distributed in their spoofing attempts to avoid being trivially detectable.

C&C communication does not require the use of the DNS, but it does provide numerous benefits: the DNS is cheap (or free), effective, and is expected by many malware kits. Peer-to-peer (P2P) C&C schemes could be used, but are more difficult to set up and maintain, and can often be too noisy for certain kinds of malicious activity (see Section 7.1.2). Direct IP connections could also be used, but at the detrimental cost of reduced agility. Simply blocking the IP addresses used for C&C by a botnet would be sufficient to permanently sever the connection between botmaster and their infected clients.

7.1.2 Evading APT Infrastructure Detection

The two primary drawbacks to the technique for detecting APTs as described in Chapter 6 are feature evasion and using C&C servers that do not rely on the DNS to bootstrap their connections. With respect to the first, we believe our features are robust enough that evasion would be detrimental to the attack (see Table 22). For the second, non-DNS C&C are either brittle or very noisy, both of which are undesirable qualities in an APT campaign.

Feature Robustness While the features used in `ghost&rae` could be altered by APT actors, we argue in most cases it would open up their infrastructure to detection by more traditional detection systems and may leave a convincing attribution trail that could lead to legal and geopolitical consequences. We believe these potential ramifications are sufficient to prevent substantial evasion from our current modeling. For example, the On/Off domain name behavior's primary purpose is to "lie low" in sensitive networks, and while malicious software can function without this it is likely to draw unwanted attention to APT infections. The use of compromised machines could be replaced with more traditional paid-for hosting infrastructure, but this introduces a money trail, which would aid law enforcement when hunting down the perpetrators. Switching away from compromised hosting could also lessen the usefulness of the infrastructure distribution features, but at the cost of having centralized points of failure for the APT campaign. Client lookup distribution features are unlikely to change, as APT threats are targeted by definition. Clever attackers could inject binaries into non-targeted networks to alter these features, but this would substantially raise their profile and potentially lead to earlier detection. In summary, while APT actors could attempt to evade the features, it opens the risk of even earlier detection and seriously damaging geopolitical consequences if successful attribution were performed.

Non-DNS C&Cs `ghost&rae` detects DNS-based APT infrastructure, but APT actors may use other means, such as directly connecting to hosts by IP [65], or by using a peer-to-peer (P2P) based C&C scheme [143]. In the former case, APT actors lose tremendous agility without the DNS at their disposal. Furthermore, in instances where the hosting infrastructure uses compromised machines as stepping stones, use of domain names is arguably necessary since these machines may become unavailable at any moment. The use of the DNS buys tremendous agility at little to no cost. For the latter case, P2P C&C schemes have become popular in traditional botnet C&Cs, likely due to their increased robustness against comprehensive takedown. However, P2P schemes often require regular communication to maintain the state of the overlay network, which introduces substantial noise. For example, the popular ZeroAccess [143] malware uses a P2P C&C channel and the initial list of peers is contained within the binary. After connecting to one of the initial peers, it updates its view of the network to maintain its connection. If the binary's initial list of peers are no longer active, the binary can no longer connect to the P2P network. This is unlikely to work well in the APT case where connections are infrequent.

7.2 Chasing Agile Attackers

While each component of the system is resilient to evasion, it would be foolhardy to assume clever and agile attackers will never be able to avoid a successful takedown we initiate. The primary benefit of the system in this dissertation is automation; condensing the largely manual process to perform a takedown before can now be done primarily automatically.

To show the benefit of such a system even against clever attackers, we show the expected running time from start to finish in Equation 6. We show that the running time is short enough that pace can be kept with evading attackers. We show the time

Table 22: Summary of modeling feature evasion consequences and the relative out-of-bag feature family importance (larger is better; see [96, 20] for details) (top) and evading the DNS (bottom). To summarize, most of the features contribute equitably to detection and removing any one of them is unlikely to completely negate `ghost&rae`'s effectiveness.

Feature Evasion	**Consequence**	**Importance**
Stop On/Off	Noisier at the network's edge	227.97
Reduce Infrastructure Distribution	Reduces plausible deniability; increases ease of attribution	172.92
Increase Client Lookup Distribution	Noisier at global level; detectable by traditional detectors	229.98
Stop using DynDNS/E2LD children	Increase financial cost	160.92
Fewer bogon resolutions	Noisier (like Stop On/Off)	75.27
DNS Evasion	**Consequence**	
Direct IP connections (no DNS)	Greatly reduces agility; suspicious	
P2P-based C&C (no DNS)	Greatly increases noise; already targeted in enterprise/government networks; likely requires non-targeted infections for overlay network	

it took to perform the postmortem takedowns described in Chapter 5.2 as well as the takedown recommendations described in Chapter 5.3. In each case, the running time is sufficiently short that attackers will not have substantial time to continue evading a takedown, which will likely be successful in the long run.

The total running time for performing a full takedown postmortem or a takedown recommendation analysis is T as defined by:

$$T = T_{IE} + T_{MI} + T_G \tag{6}$$

where T_{MI} is the time to perform malware interrogation on the botnet, T_{IE} is the time it takes to perform infrastructure enumeration, and T_G is the running time for modeling and clustering the infrastructure to detect likely collateral damage in the form of APT infrastructure. Each sub-equation is defined below:

$$T_{IE} = 2\alpha P \tag{7}$$

Equation 7 represents the time to fully enumerate the infrastructure of a botnet's criminal network. P represents the time to perform a full pass of the passive DNS database to identify related infrastructure and α represents the number passes that must be performed until the infrastructure converges. Two passes over the passive DNS database must be made in order to enumerate infrastructure (see Figure 6); once to identify related historic IP addresses and a second time to identify related historic domain names. Recall Figure 14 that shows the process to perform a takedown recommendation. If additional infrastructure is identified through malware interrogation, an additional pass over the passive DNS is needed to continue to expand our knowledge of the infrastructure.

Performing a pass over the passive DNS database can be done with either bulk or individual requests. In a bulk request, the running time is $\mathcal{O}(1)$ with respect to the number of domains but a single bulk request runs on the order of minutes, while with individual requests the running time is $\mathcal{O}(n)$ in the number of domain names that must be queried but a single request finishes in seconds. For ease of presentation, we only consider bulk requests but in an operational environment individual requests would be made where time would be saved. As we show, however, most infrastructure can be enumerated in a single pass.

$$T_{MI} = \frac{m \times 7t}{\mathcal{M}} \tag{8}$$

Equation 8 is the time to fully interrogate the malware of the botnet to extract any additional network endpoints that must be disabled, as well as any potential backup C&C behavior included in the botnet's infrastructure. This is limited by the number of machines we have available for performing malware analysis, $\mathcal{M}$, the duration of the control malware analysis run, t, and the number of malware samples related to the botnet's infrastructure m. Recall that each malware sample must be run five times, two of which for duration $2t$, in order to understand the malware's backup plan (see Chapter 3.3.3).

$$T_G = P + G + R \tag{9}$$

Equation 9 represents the time to extract features from, classify, and cluster the domain names currently being considered for a takedown as being APT-related or not. Recall that APT domains often use compromised machines for C&C servers so such takedowns are likely to include benign domains if network operators are not diligent. The APT modeling features as described in Chapter 6.3.2.1 require passive DNS evidence that also take time P as described above. To generate the features, only a single pass over the passive DNS data is required. The remaining time is to perform the modeling as described in Chapter 6.3.2.2, G, and to generate clusters as described in Chapter 6.3.3.1, R. As the running times G and R are difficult to quantify reliably in terms of the input size, we simply provide an upper bound for each when modeling and clustering one million resource records.

Bound and Free Variables Throughout the evaluation of the systems, the following variables were bound:

- $M = 512$ virtual machines for malware interrogation.
- $t = 3$ minutes for the control malware execution timeout. Note malware may terminate earlier of its own accord, but malware will run for at most three minutes.
- $G = 30$ minutes to perform modeling on one million resource record feature vectors.
- $R = 240$ minutes to cluster one million classified resource records.

Table 23: Timings for end-to-end run of postmortem studies.

Takedown	α	m	T (in hours)
Kelihos	1	199	5.68
Zeus	1	2,428	7.20
Nitol	1	12,745	14.25

- $P = 21$ minutes to make a single, full pass over the passive DNS database.

This leaves two free variables to compute for each postmortem or takedown recommendation: α and m the number of runs of the system until convergence is achieved and the number of malware samples to be analyzed, respectively.

7.2.1 Postmortem Running Time

We revisit the takedown postmortems presented in Chapter 5.2 for three botnet takedowns: Kelihos, Zeus, and Nitol (3322.org). The timing information is summarized in Table 23

7.2.2 Takedown Recommendation Running Time

Table 24: Timings for end-to-end run for takedown recommendations.

ID	α	m	T (in hours)
1	1	6,431	9.93
2	1	24,212	22.10
3	1	43	5.57
4	1	1	5.55
5	1	59	5.58
6	1	344	5.78
7	1	999	6.22
8	1	94,248	69.97
9	1	2	5.55
10	1	810	6.10
11	1	3	5.55
12	1	75	5.60
13	1	220	5.70
14	1	92	5.60
15	1	129,431	94.02
16	1	4,978	8.95
17	1	1	5.55
18	1	71	5.58
19	1	1,325	6.45
20	1	3,305	7.80
21	1	4,421	8.57
22	1	323	5.77
23	1	1	5.55
24	1	552	5.92
25	1	596	5.95
26	1	7,311	10.53
27	1	303	5.75
28	1	33	5.57
29	1	5,517	9.32
30	1	91,355	67.98
31	1	238	5.70
32	1	34	5.57
33	1	87	5.60
34	1	42	5.57
35	1	8	5.55
36	1	180	5.67
37	1	59,062	45.92
38	1	3,391	7.87
39	1	502	5.88
40	1	53	5.58
41	1	34,062	28.83
42	1	148	5.65
43	1	1,021	6.23
44	1	52	5.58
45	1	15	5.55
Average	1	10,346	12.77

CHAPTER VIII

CONCLUSIONS

8.1 Overall Contributions and Summary

To automatically enumerate infrastructure in preparation for a takedown three tasks must be performed: malware-based infrastructure enumeration, network-based infrastructure enumeration, and to identify potential collateral damage. Using the output from these tasks, we evaluate the success of historic takedowns and recommend actions for a future takedown.

Malware Infrastructure Enumeration We built and evaluated `gza`, which interrogates malware to reveal alternative execution plans of malware and classify instances where the backup behavior is more elaborate, e.g., DGA- or P2P-based C&C scheme. `gza` can coerce hundreds of thousands of likely malicious domain names from malware that never appeared on public blacklists after our study and that sophisticated backup C&C techniques, such as DGA or P2P, can be reliably detected by playing games with malware. Both are necessary to perform a successful takedown.

`gza` is available as open-source software [1] as a module to the popular malware analysis framework "Cuckoo Sandbox."

Network Infrastructure Enumeration We built `deck` to enumerate malicious infrastructure based on an initial set of *seed domains*, i.e., initial knowledge of a set of infrastructure, by analyzing the *passive DNS* history of the seed domains. `deck` builds and visualizes graph representations of malicious infrastructure and analyzes them with respect to performing comprehensive takedowns using network analysis

[1] `https://github.com/ynadji/cuckoo/tree/gza`

measures. We found that in many cases criminal networks can be disabled by deregistering as few as five domain names. In more sophisticated networks, targeting the C&C servers themselves can cause substantial damage to the infrastructure. In one case, disabling 20% of a criminal network's hosts would reduce the overall volume of successful connections by 70%, suggesting that certain "critical" nodes are more important to the infrastructure.

`deck`'s visualization of criminal networks has been implemented into the product offering of Damballa, Inc.

Measuring & Recommending Takedowns We built `rza`—a combination of `gza` and `deck`—to measure the success of historic takedowns and use a similar methodology to recommend how to perform a takedown and what precautionary measures must be taken to ensure the takedown is successful. We find that several historic takedowns were performed without fully enumerating the criminal infrastructure, allowing malicious activity to continue unfettered. We also show many threats active at the time of the study can be disabled easily by only "sinkholing" or disabling a handful of domain names. Finally, many of the studied takedowns caused serious *collateral damage* to legitimate customers; this motivates the need to build tools to identify network assets likely shared between benign users and Internet miscreants to ensure a takedown action does not cause undue harm.

The output of `rza` for several botnets has been given to the FBI to inform on the backup behavior of malware and the feasibility of taking down their criminal infrastructure.

Collateral Damage—Advanced Threats We presented `ghost&rae`, a system to model "manual APT" infrastructure that commonly resides on compromised hosts alongside benign domains. After classifying these domains, we cluster them to separate malicious APT domains from benign domains to prevent them from being taken down and disabled as collateral damage. We show that `ghost&rae` can model APT domains with high true positive and low false positive rates. While this is only one class of collateral damage to detect, it represents an interesting case as it is a common source of false positives for existing reputation systems [4, 14].

`ghost&rae` is currently being tested for production deployment at Damballa, Inc.

8.2 Lessons Learned

Throughout the process of writing the dissertation, we have identified some key lessons learned with respect to botnet takedowns and APT infrastructure.

8.2.1 Botnet Takedowns

Botnets have prevailed despite concerted efforts to both clean infected machines as well as disable the C&C infrastructure. This is partially due to the lack of automated enumeration techniques as discussed and tackled in detail in this dissertation. However, there are other pitfalls the security community must be cautious of when performing takedowns.

We have shown that many real-world botnets can be disabled by revoking or sinkholing fewer than dozens of domain names. These botnets are typically much smaller than those targeted in high-profile takedown cases, but nonetheless they can be disabled easily. However, even in successful takedowns, such as Kelihos 5.2.2, we see that the botnets do not always go away forever. While the first variant of Kelihos was successfully disabled, the perpetrators were not initially arrested and a new variant quickly began to spread.

This begs the question: is a takedown that leaves the botmasters free to infect another day beneficial for the security community? On one hand, it puts a temporary stop to profits from cybercrime, but on the other hand it forces attackers to build more resilient infrastructure. Further study is needed to quantify this cost, and more importantly, perform attribution to assist law enforcement in arresting the botmasters.

Finally, sophisticated C&C infrastructure, such as DGA/P2P-based schemes, must be handled appropriately to ensure a successful takedown. In the case of DGAs, carefully reversing the DGA and working with the registrars as appropriate is necessary to successfully perform a takedown. The case of P2P is harder still; while prior work [142] describes techniques to disable these botnets, one has yet to be disabled. The main conclusion to be drawn is if the security community does not have a plan of action to handle sophisticated C&C infrastructure, it is better to *not* attempt a takedown at all.

8.2.2 Manual APT Infrastructure

While studying manual APT infrastructure that was identified by `ghost&rae`, we identified some key takeaways and avenues for further research surrounding APTs. First, malware is not a strong indicator of APT threats. Second, the features described in Chapter 6 are likely robust since they describe measurable properties of APT attacks. Finally, a taxonomy of APT attacks is needed to further understand how detection and remediation of these threats can be performed in a more general manner.

While some APT threats use "advanced" malware, most APTs use commodity malware. This is corroborated by the industry reports on APTs we studied, as well as our findings in Section 6.4, where the highest confidence APT-related infrastructure `ghost&rae` identified were used by generic trojans. This suggests that not only is the 'A' in APT incorrect, additional work on detecting these threats should move away from focusing on the malware samples currently in use by attackers.

Choosing robust features is paramount to building a successful machine learning model. Not only are the features we designed robust in practice (see Section 7.1.2), but the intuition for them are based on measurable properties of APT attacks, e.g., they are targeted and persistent, rather than properties surrounding the intent of the attack, e.g., stealing intellectual property, or properties that are difficult to reliably detect, e.g., perpetrated by a nation-state.

While `ghost&rae` provides a good starting point for detection, building a taxonomy of APTs seen in the wild would further improve the state-of-the-art. The assumptions made for "manual" APTs are likely not true for *all* APTs, but this cannot be known unless a comprehensive taxonomy is created. This would not only improve future detection systems, but expand our knowledge of a particularly nefarious class of threat.

REFERENCES

[1] Abu Rajab, M., Zarfoss, J., Monrose, F., and Terzis, A., "A multifaceted approach to understanding the botnet phenomenon," *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006.

[2] Adams, S., "Conficker Windows virus infects 15 million PCs," 2009.

[3] Alexa, "Top sites." `http://www.alexa.com/topsites`, (Retrieved) March 2011.

[4] Antonakakis, M., Perdisci, R., Dagon, D., and Lee, W., "Building a Dynamic Reputation System for DNS," in *Proceedings of the USENIX Security Symposium*, 2010.

[5] Antonakakis, M., Perdisci, R., Dagon, D., and Lee, W., "Building a dynamic reputation system for DNS," in *Proceedings of the 19th USENIX Security Symposium*, 2010.

[6] Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., and Dagon, D., "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware," in *Proceedings of 21th USENIX Security Symposium (USENIX Security '12)*, 2012.

[7] Ask, Merete and Bondarenko, Petro and Rekdal, John Erik and Nordbø, André and Bloemerus, Pieter and Piatkivskyi, Dmytro, "Advanced Persistent Threat (APT) Beyond the hype," tech. rep. `https://andynor.net/static/fileupload/434/S2_NetwSec_Advanced_Persistent_Threat.pdf`.

[8] BalaBit, "BalaBit eCSI Report," tech. rep., 2015. `https://pages.balabit.com/rs/balabititsecurity/images/BalaBit-eCSI-magazine-201503-10-13.pdf`.

[9] Balduzzi, M., Ciangaglini, V., and McArdle, R., "Targeted attacks detection with spunge," in *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on*, pp. 185–194, IEEE, 2013.

[10] Balzarotti, D., Cova, M., Karlberger, C., Kruegel, C., and Kirda, E., "Efficient detection of split personalities in malware," in *Proceedings of the Symposium on Network and Distributed System Security*, 2010.

[11] Bastian, M., Heymann, S., and Jacomy, M., "Gephi: An Open Source Software for Exploring and Manipulating Networks," in *International AAAI Conference on Weblogs and Social Media*, 2009.

[12] Bates, T., Smith, P., and Huston, G., "CIDR report bogons." http://www.cidr-report.org/bogons/.

[13] Beth E. Binde, Russ McRee, Terrence J. O'Connor, "Assessing Outbound Traffic to Uncover Advanced Persistent Threat," tech. rep. https://www.sans.edu/student-files/projects/JWP-Binde-McRee-OConnor.pdf.

[14] Bilge, L., Kirda, E., Kruegel, C., and Balduzzi, M., "EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis," in *Network and Distributed System Security Symposium (NDSS)*, 2011.

[15] Bilge, L., Kirda, E., Kruegel, C., and Balduzzi, M., "EXPOSURE: Finding malicious domains using passive DNS analysis," in *Proceedings of the Symposium on Network and Distributed System Security*, Jan 2011.

[16] Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., and Kruegel, C., "Disclosure: Detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 129–138, ACM, 2012.

[17] Biondi, P., "Scapy." http://www.secdev.org/projects/scapy/, (Retrieved) March 2011.

[18] Blondel, V., Guillaume, J., Lambiotte, R., and Lefebvre, E., "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[19] Boscovich, R. D., "Microsoft Reaches Settlement with Defendants in Nitol Case," 2012.

[20] Breiman, L., "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] Brin, S. and Page, L., "The anatomy of a large-scale hypertextual web search engine," in *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, (Amsterdam, The Netherlands, The Netherlands), pp. 107–117, Elsevier Science Publishers B. V., 1998.

[22] BRONK, C., "Risk–intelligent governance in the age of cyberthreats," *Available at SSRN 2270853*, 2013.

[23] Brumley, D., Hartwig, C., Liang, Z., Newsome, J., Song, D., and Yin, H., "Automatically identifying trigger-based behavior in malware," *Botnet Detection*, pp. 65–88, 2008.

[24] Burnette, S., "Tucows Notice of Breach of Registrar Accreditation Agreement.".

[25] Bursztein, E. and Mitchell, J. C., "Using strategy objectives for network security analysis," *Information Security and Cryptology*, Jan 2011.

[26] Caballero, J., Grier, C., and Kreibich, C., "Measuring Pay-per-Install: The Commoditization of Malware Distribution," in *Proceedings of the USENIX Security Symposium*, 2011.

[27] Caballero, J., Poosankam, P., Kreibich, C., and Song, D., "Dispatcher: Enabling active botnet infiltration using automatic protocol reverse-engineering," *ACM Conference on Computer and Communications Security*, vol. 16th, 2009.

[28] Carroll, T. and Grosu, D., "A game theoretic investigation of deception in network security," *Security and Communication Networks*, Jan 2009.

[29] Chen, X., Andersen, J., Mao, Z. M., Bailey, M., and Nazario, J., "Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware," in *Proceedings of the International Conference on Dependable Systems and Networks DSN*, 2008.

[30] Cheswick, B., "An evening with berferd in which a cracker is lured, endured, and studied," in *Proceedings of the USENIX Security Symposium*, Jan 1990.

[31] Cho, C., Caballero, J., and Grier, C., "Insights from the inside: A view of botnet management from infiltration," in *Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.

[32] Christin, N., Yanagihara, S. S., and Kamataki, K., "Dissecting one click frauds," in *Proceedings of the 17th ACM Conference on Computer and Communiations Security (CCS)*, 2010.

[33] Christodorescu, M., Jha, S., Seshia, S., Song, D., and Bryant, R., "Semantics-aware malware detection," *IEEE Symposium on Security and Privacy*, pp. 32–46, 2005.

[34] CIOL.com, "Attacks to steal sensitive data from Defense Metallurgical Research Laboratory got detected." `http://www.ciol.com/attempt-steal-sensitive-data-defense-metallurgical-research-laboratory/`.

[35] Clauset, A. and Newman, M., "Finding community structure in very large networks," in *Physical review E*, 2004.

[36] Cohen, F. and Koike, D., "Misleading attackers with deception," in *Proceedings of the 2004 IEEE Workshop on Information Assurance*, Jan 2004.

[37] Collins, M., Shimeall, T., Faber, S., Janies, J., Weaver, R., and Shon, M. D., "Predicting future botnet addresses with uncleanliness," in *Proc. of IMC*, 2007.

[38] Conficker Working Group, "Conficker Working Group: Lessons Learned," 2011. `http://www.confickerworkinggroup.org/wiki/uploads/Conficker_Working_Group_Lessons_Learned_17_June_2010_final.pdf`.

[39] Correa, A. D., "Malware patrol." `http://www.malware.com.br/`.

[40] Correa, A. D., "Malware patrol." `http://malwarepatrol.com/`, 2010.

[41] Cova, M., Leita, C., Thonnard, O., Keromytis, A., and Dacier, M., "An analysis of rogue AV campaigns," in *Recent Advances in Intrusion Detection*, 2010.

[42] CrowdStrike, Inc., "CrowdStrike Global Threat Report (2013 Year in Review)," tech. rep., 2013. `http://www.crowdstrike.com/sites/all/themes/crowdstrike2/css/imgs/platform/CrowdStrike_Global_Threat_Report_2013.pdf`.

[43] Cymru, T., "Bogons." `http://www.cymru.com/Documents/bogon-bn-nonagg.txt`, 2010.

[44] Dan Kelly and Tom Lancaster, "OrcaRAT—A whale of a tale." `http://pwc.blogs.com/cyber_security_updates/2014/10/orcarat-a-whale-of-a-tale.html`.

[45] Dinaburg, A., Royal, P., Sharif, M., and Lee, W., "Ether: Malware analysis via hardware virtualization extensions," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, Jan 2008.

[46] dn1nj4, "RBN "Rizing"," tech. rep., Shadowserver.org, 2008.

[47] DNS-BH, "Malware prevention through DNS redirection." `http://www.malwaredomains.com`.

[48] DNS-BH, "Malware prevention through DNS redirection (black hole DNS sinkhole)." `http://www.malwaredomains.com`, 2010.

[49] dnsbl.abuse.ch, "dnsbl.abuse.ch." `http://dnsbl.abuse.ch/`.

[50] dnsbl.abuse.ch, "dnsbl.abuse.ch." `http://dnsbl.abuse.ch`, 2010.

[51] dnswl, "DNS whitelist - protect against false positives." `http://www.dnswl.org`, (Retrieved) March 2011.

[52] Downes, L., "Requiem for Failed UN Telecom Treaty: No One Mourns the WCIT," 2012.

[53] Duda, R., Hart, P., and Stork, D., *Pattern Classification.* Wiley-Interscience, 2nd ed., 2000.

[54] F-Secure, "Preemptive Blocklist and More Downadup Numbers," 2009.

[55] Falliere, N., Murchu, L. O., and Chien, E., "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, 2011.

[56] Farsight Security, Inc., "DNSDB." https://www.dnsdb.info/.

[57] Farsight Security, Inc., "SIE/Farsight Security's DNSDB," 2013. https://www.dnsdb.info/.

[58] Federal Communications Commission, "CSRIC Adopts Recommendations to Minimize Three Major Cyber Threats," tech. rep., Federal Communications Commission, 2012.

[59] Felegyhazi, M., Kreibich, C., and Paxson, V., "On the potential of proactive domain blacklisting," in *Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats*, 2010.

[60] Gath, I. and Geva, A. B., "Unsupervised optimal fuzzy clustering," *Pattern Analysis and Machine Intelligence*, 1989.

[61] Geers, K., Kindlund, D., Moran, N., and Rachwald, R., "World War C: Understanding Nation-State Motives Behind Today's Advanced Cyber Attacks," tech. rep., FireEye Labs, 2013. http://www.fireeye.com/resources/pdfs/fireeye-wwc-report.pdf.

[62] Goguen, N., "No-ip's formal statement on microsoft takedown," 2014. http://www.noip.com/blog/2014/06/30/ips-formal-statement-microsoft-takedown/.

[63] Goncharov, M., "Russian Underground 101," tech. rep., Trend Micro Incorporated, 2012.

[64] GReAT, "The Icefog APT: A Tale of Cloak and Three Daggers," Sept. 2013. https://www.securelist.com/en/blog/208214064/The_Icefog_APT_A_Tale_of_Cloak_and_Three_Daggers.

[65] Guarnieri, C., "ByeBye Shell and the targeting of Pakistan," Aug. 2013. https://community.rapid7.com/community/infosec/blog/2013/08/19/byebye-and-the-targeting-of-pakistan.

[66] Guo, F., Ferrie, P., and Chiueh, T., "A study of the packer problem and its solutions," in *Proceedings of the Symposium on Recent Advances in Intrusion Detection (RAID 2008)*, pp. 98–115, 2008.

[67] Hardy, S., Crete-Nishihata, M., Kleemola, K., Senft, A., Sonne, B., Wiseman, G., Gill, P., and Deibert, R. J., "Extended Analysis: Cluster Analysis," tech. rep., 2014. https://targetedthreats.net/media/2.2%20Extended%20Analysis-Cluster.pdf.

[68] Hardy, S., Crete-Nishihata, M., Kleemola, K., Senft, A., Sonne, B., Wiseman, G., Gill, P., and Deibert, R. J., "Targeted threat index: Characterizing and quantifying politically-motivated targeted malware," in *23rd USENIX Security Symposium (USENIX Security 14)*, (San Diego, CA), pp. 527–541, USENIX Association, Aug. 2014.

[69] Harris, M., "Spammers recovering from McColo shutdown," 2009. `http://www.techradar.com/news/internet/spammers-recovering-from-mccolo-shutdown-591118`.

[70] Higgins, K. J., "Microsoft Intercepts 'Nitol' Botnet And 70,000 Malicious Domains," 2012.

[71] Hofmeyr, S. and Forrest, S., "An immunological model of distributed detection and its application to computer security," *University of New Mexico*, Jan 1999.

[72] Holz, T., Engelberth, M., and Freiling, F., "Learning more about the underground economy: A case-study of keyloggers and dropzones," in *Computer Security–ESORICS 2009*, 2010.

[73] Holz, T., Gorecki, C., and Freiling, F., "Detection and mitigation of fast-flux service networks," *Proceedings of the 15th ...*, Jan 2008.

[74] Intel, "Not Your Average Cybercriminal: A Look at the Diverse Threats to the Financial Services Industry," Sept. 2013.

[75] Internet Corporation for Assigned Names and Numbers, "Uniform Domain Name Dispute Resolution Policy," tech. rep., 1999.

[76] Internet Corporation for Assigned Names and Numbers, "Uniform Rapid Suspension System," tech. rep., 2012.

[77] Internet Systems Consortium, "Security Information Exchange Portal," 2010. `https://www.isc.org/blogs/join-the-global-passive-dns-pdns-network-today-gain-effective-tools-to-fight`

[78] James T. Bennett, "The Mutter Backdoor: Operation Beebus with New Targets." `http://www.fireeye.com/blog/technical/malware-research/2013/04/the-mutter-backdoor-operation-beebus-with-new-targets.html`.

[79] Jesa Golez, "Trend Micro—Threat Encyclopedia." `http://www.trendmicro.com/vinfo/us/threat-encyclopedia/malicious-url/2911`.

[80] Jiang, J., Liang, J., Li, K., Li, J., Duan, H., and Wu, J., "Ghost Domain Names: Revoked Yet Still Resolvable," in *Network and Distributed System Security Symposium (NDSS)*, 2012.

[81] John, J., Moshchuk, A., Gribble, S., and Krishnamurthy, A., "Studying spamming botnets using botlab," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, pp. 291–306, 2009.

[82] Kang, M., Poosankam, P., and Yin, H., "Renovo: a hidden code extractor for packed executables," *Proceedings of the 2007 ACM workshop on Recurring malcode*, Jan 2007.

[83] Kang, M., Yin, H., Hanna, S., McCamant, S., and Song, D., "Emulating emulation-resistant malware," *Proceedings of the 1st ACM workshop on Virtual machine security*, pp. 11–22, 2009.

[84] Kaspersky Research Lab (GREAT), "The 'Icefog' APT: A Tale of Cloak and Three Daggers," tech. rep., Kaspersky, 2013. `http://www.securelist.com/en/downloads/vlpdfs/icefog.pdf`.

[85] Katz, S., "Hierarchical mesh decomposition using fuzzy clustering and cuts," in *ACM SIGGRAPH 2003 Papers*, 2003.

[86] Konte, M., Feamster, N., and Jung, J., "Fast flux service networks: Dynamics and roles in hosting online scams," tech. rep., 2008.

[87] Konte, M., Feamster, N., and Jung, J., "Dynamics of online scam hosting infrastructure," in *Passive and Active Network Measurement*, 2009.

[88] Kothari, K. and Wright, M., "Mimic: An active covert channel that evades regularity-based detection," *Computer Networks*, vol. 57, no. 3, pp. 647 – 657, 2013.

[89] Krebs, B., "Major Source of Online Scams and Spams Knocked Offline," 2008. `http://voices.washingtonpost.com/securityfix/2008/11/major_source_of_online_scams_a.html`.

[90] Krebs, B., "Spam Volumes Drop by Two-Thirds After Firm Goes Offline," 2008. `http://voices.washingtonpost.com/securityfix/2008/11/spam_volumes_drop_by_23_after.html`.

[91] Krebs, B., "Mariposa Botnet Authors May Avoid Jail Time," 2010. `http://krebsonsecurity.com/2010/03/mariposa-botnet-authors-may-avoid-jail-time/`.

[92] Langner, R., "Stuxnet: Dissecting a cyberwarfare weapon," *Security & Privacy, IEEE*, vol. 9, no. 3, pp. 49–51, 2011.

[93] Lavasoft Security Center, ""Letter to President Obama"." `http://www.lavasoft.com/mylavasoft/securitycenter/whitepapers/%E2%80%9Cletter-to-president-obama%E2%80%9D`.

[94] LEONTIADIS, N., MOORE, T., and CHRISTIN, N., "Measuring and analyzing search-redirection attacks in the illicit online prescription drug trade," in *Proceedings of the USENIX Security Symposium*, August 2011.

[95] LEYDEN, J., "State-backed hackers: You think you're so mysterious, but you're really not – report," Feb. 2013. `http://www.theregister.co.uk/2013/10/02/nation_state_cyberattack/`.

[96] LIAW, A. and WIENER, M., "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[97] LIST, M. D., "Malware domain list." `http://www.malwaredomainlist.com`, 2010.

[98] LIU, H., LEVCHENKO, K., FÉLEGYHÁZI, M., KREIBICH, C., MAIER, G., VOELKER, G. M., and SAVAGE, S., "On the effects of registrar-level intervention," in *LEET'11: Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, 2011.

[99] LU, L., YEGNESWARAN, V., PORRAS, P., and LEE, W., "BLADE: an attack-agnostic approach for preventing drive-by malware infections," in *Proceedings of the 17th ACM Conference on Computer and Communiations Security (CCS 2010)*, Jan 2010.

[100] LU, L., YEGNESWARAN, V., PORRAS, P., and LEE, W., "BLADE: an attack-agnostic approach for preventing drive-by malware infections," in *Proceedings of the 17th ACM Conference on Computer and Communiations Security (CCS 2010)*, 2010.

[101] MALC0DE, "Malc0de DNS blacklist." `http://malc0de.com/bl/`.

[102] MALC0DE, "Malc0de DNS blacklist." `http://malc0de.com`, 2010.

[103] MALWARE DOMAIN LIST, "Malware domain list." `http://www.malwaredomainlist.com/`.

[104] MANDIANT, "APT1," tech. rep., 2013. `http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf`.

[105] MANDIANT, "M-Trends," tech. rep., 2013. `https://dl.mandiant.com/EE/library/M-Trends_2013.pdf`.

[106] MARTIGNONI, L., STINSON, E., FREDRIKSON, M., and JHA, S., "A layered architecture for detecting malicious behaviors," *Recent Advances in ...*.

[107] MATROSOV, A., "TDSS part 1 through 4." `http://resources.infosecinstitute.com/tdss4-part-1/`, 2011.

[108] McCoy, D., Pitsillidis, A., Jordan, G., Weaver, N., Kreibich, C., Krebs, B., Voelker, G. M., Savage, S., and Levchenko, K., "Pharmaleaks: Understanding the business of online pharmaceutical affiliate programs," in *21st Usenix Security Symposium (USENIX 2012)*, 2012.

[109] McMillan, R., "After takedown, botnet-linked ISP Troyak resurfaces," 2010. http://www.computerworld.com/s/article/9169118/After_takedown_botnet_linked_ISP_Troyak_resurfaces.

[110] Meyer, D., "Route Views Archive Project." http://routeviews.org/.

[111] Microsoft Corporation, "Microsoft Corporation v. Dominique Alexander Piatti; Jone Does 1-22," 2011. Virginia Eastern District Court.

[112] Microsoft Corporation, "Microsoft Corporation v. Peng Yong et. al.," 2012. Virginia Eastern District Court.

[113] Microsoft Corporation, "Microsoft v. John Does 1-39," 2012. New York Eastern District Court.

[114] Mills, E., "Microsoft halts another botnet: Kelihos," 2011. http://news.cnet.com/8301-1009_3-20112289-83/microsoft-halts-another-botnet-kelihos/.

[115] Moser, A., Kruegel, C., and Kirda, E., "Exploring multiple execution paths for malware analysis," in *Proceedings of the IEEE Symposium on Security and Privacy*, vol. 245, 2007.

[116] Mueller, M., *Ruling the root.* The MIT Press, 2004.

[117] Nadji, Y., Antonakakis, M., Perdisci, R., and Lee, W., "Understanding the prevalence and use of alternative plans in malware with network games," in *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC '11)*, 2011.

[118] Nagaraja, S. and Anderson, R., "The topology of covert conflict," in *Workshop on the Economics of Information Security (WEIS)*, 2006.

[119] Nagaraja, S., Mittal, P., Hong, C.-Y., Caesar, M., and Borisov, N., "Botgrep: finding p2p bots with structured graph analysis," in *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, (Berkeley, CA, USA), pp. 7–7, USENIX Association, 2010.

[120] Nart Villeneuve and James Bennett, "Detecting APT Activity with Network Traffic Analysis," tech. rep. http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-detecting-apt-activity-with-network-traffic-analysis.pdf.

[121] NETFILTER TEAM, "The netfilter.org "iptables" project." http://www.netfilter.org/projects/iptables/index.html, (Retrieved) March 2011.

[122] NEUGSCHWANDTNER, M. and COMPARETTI, P., "Detecting malware's failover C&C strategies with squeeze," in *Proceedings of the 27th . . .*, 2011.

[123] NEWMAN, M., *Networks: An Introduction.* Oxford University Press, USA, 2010.

[124] OPREA, A., LI, Z., YEN, T., CHIN, S., and ALRWAIS, S. A., "Detection of early-stage enterprise infection by mining large-scale log data," *CoRR*, vol. abs/1411.5005, 2014.

[125] PAGE, L., BRIN, S., MOTWANI, R., and WINOGRAD, T., "The PageRank citation ranking: Bringing order to the web.," 1999.

[126] PAXSON, V., "Bro: a system for detecting network intruders in real-time," *Computer networks*, Jan 1999.

[127] PELLEG, D. and MOORE, A. W., "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in *Seventeenth International Conference on Machine Learning*, pp. 727–734, Morgan Kaufmann, 2000.

[128] PERDISCI, R., CORONA, I., DAGON, D., and LEE, W., "Detecting malicious flux service networks through passive analysis of recursive DNS traces," *Proceedings of the 2009 . . .*, Jan 2009.

[129] PERDISCI, R., LEE, W., and FEAMSTER, N., "Behavioral clustering of HTTP-based malware and signature generation using malicious network traces," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, 2010.

[130] PFANNER, E., "Citing Internet Standoff, U.S. Rejects International Telecommunications Treaty," 2012.

[131] PHILLIP PORRAS, H. S. and YEGNESWARAN, V., "An analysis of conficker's logic and rendezvous points." http://mtc.sri.com/Conficker/, 2009.

[132] PROJECT, S., "Snort DNS/IP/URL lists." http://labs.snort.org/iplists/, 2011.

[133] PROJECT, T. S., "Spamhaus drop list." http://www.spamhaus.org/drop/drop.lasso, 2011.

[134] PROVOS, N., MAVROMMATIS, P., RAJAB, M., and MONROSE, F., "All your iframes point to us," in *Proceedings of the 17th conference on Security symposium*, 2008.

[135] Public Interest Registry (PIR), "Our New Initiatives to Combat Botnets," 2012. http://www.circleid.com/posts/20121203_our_new_initiatives_to_combat_botnets/.

[136] Raffetseder, T., Krügel, C., and Kirda, E., "Detecting system emulators," in *Information Security Conference*, pp. 1–18, 2007.

[137] Raiu, C., "NetTraveler Is Back: The 'Red Star' APT Returns With New Tricks," Sept. 2013. http://www.securelist.com/en/blog/208214039/NetTraveler_Is_Back_The_Red_Star_APT_Returns_With_New_Tricks.

[138] Ramachandran, A., Feamster, N., and Vempala, S., "Filtering spam with behavioral blacklisting," in *Proceedings of the 14th ...*, 2007.

[139] Ramachandran, A. and Feamster, N., "Understanding the network-level behavior of spammers," in *ACM SIGCOMM Computer ...*, 2006.

[140] Report, C., "CIDR report bogons." http://www.cidr-report.org, 2011.

[141] Riden, J., "How fast-flux service networks work." http://www.honeynet.org/node/132, 2008.

[142] Rossow, C., Andriesse, D., and Werner, T., "P2PWNED: Modeling and Evaluating the Resilience of Peer-to-Peer Botnets," in *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P 2013)*, 2013.

[143] Rossow, C., Andriesse, D., Werner, T., Stone-Gross, B., Plohmann, D., Dietrich, C. J., and Bos, H., " P2PWNED: Modeling and Evaluating the Resilience of Peer-to-Peer Botnets ," in *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)* , (San Francisco, CA), May 2013.

[144] Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H., and Van Steen, M., "Prudent practices for designing malware experiments: Status quo and outlook," in *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 65–79, IEEE, 2012.

[145] Roveta, F., Mario, L. D., Maggi, F., Caviglia, G., Zanero, S., and Ciuccarelli, P., "BURN: Baring Unknown Rogue Networks," in *VizSec*, 2011.

[146] Rowe, N., Custy, E., and Duong, B. T., "Defending cyberspace with fake honeypots," *Journal of Computers*, Jan 2007.

[147] Royal, P., Halpin, M., and Dagon..., D., "Polyunpack: Automating the hidden-code extraction of unpack-executing malware," ..., Jan 2006.

[148] Rutkowska, J., "Red pill... or how to detect VMM using (almost) one CPU instruction." http://invisiblethings.org/papers/redpill.html, 2004.

[149] Santiago Cortes, "Backdoor.Sacto." http://www.symantec.com/security_response/writeup.jsp?docid=2013-071613-4738-99&tabid=2.

[150] Seculert, Inc., "Combating Advanced Persistent Threats through Detection," tech. rep. http://info.seculert.com/hs-fs/hub/237610/file-29639693-pdf/White_Papers/Seculert-Combating-Ad.

[151] Sharif, M., Lanzi, A., Giffin, J., and Lee, W., "Rotalume: A tool for automatic reverse engineering of malware emulators,"

[152] Sharif, M., Lanzi, A., Giffin, J., and Lee, W., "Automatic reverse engineering of malware emulators," *2009 30th IEEE Symposium on Security and Privacy*, pp. 94–109, 2009.

[153] Sharif, M., Lanzi, A., Giffin, J., and Lee, W., "Impeding malware analysis using conditional code obfuscation," in *Proceedings of the Symposium on Network and Distributed System Security*, Jan 2008.

[154] Smith, R., "tesseract-ocr." https://code.google.com/p/tesseract-ocr/.

[155] Snort Labs, "Snort DNS/IP/URL lists." http://labs.snort.org/iplists/.

[156] Sophos, "Troj/Agent-QYN." http://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Agent-QYN/detailed-analysis.aspx.

[157] Sophos, "Troj/Agent-RHN." http://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Agent-RHN/detailed-analysis.aspx.

[158] Sophos, "Troj/Agent-UVN." http://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Agent-UVN/detailed-analysis.aspx.

[159] Sophos, "Troj/Cordmix-A." http://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Cordmix-A/detailed-analysis.aspx.

[160] Sophos, "Troj/Cordmix-B." http://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Cordmix-B/detailed-analysis.aspx.

[161] SpamHaus, "drop.lasso." http://www.spamhaus.org/drop/drop.lasso.

[162] SpyEye Tracker, "SpyEye tracker." https://spyeyetracker.abuse.ch/.

[163] spyeyetracker.abuse.ch, "Spyeye tracker." https://spyeyetracker.abuse.ch, 2010.

[164] Staniford, S., Paxson, V., and Weaver, N., "How to 0wn the Internet in your Spare Time," in *Proceedings of the USENIX Security Symposium*, 2002.

[165] Stewart, J. and Jackson, D., "Secrets of the Comfoo Masters," July 2013. http://www.secureworks.com/cyber-threat-intelligence/threats/secrets-of-the-comfoo-masters/.

[166] Stone-Gross, B., Kruegel, C., Almeroth, K., Moser, A., and Kirda, E., "Fire: Finding rogue networks," in *ACSAC*, 2009.

[167] Stranger, P., McQuaid, J., Burn, S., Glosser, D., Freezel, G., Thompson, B., and Rogofsky, W., "Top 50 Bad Hosts and Networks," *Tech Report*.

[168] Symantec, Inc., "Symantec Internet Security Threat Report," tech. rep., 2010. http:////www.symantec.com/business/theme.jsp?themeid=threatreport.

[169] Tan, P.-N., Steinbach, M., and Kumar, V., *Introduction to Data Mining*. Addison-Wesley Longman Publishing, 2005.

[170] Tan, P.-N., Steinbach, M., and Kumar, V., *Introduction to Data Mining*. Addison Wesley, 2006.

[171] TaoSecurity, "What Is APT and What Does It Want?." http://taosecurity.blogspot.com/2010/01/what-is-apt-and-what-does-it-want.html.

[172] Team Cymru, "Bogons." https://www.team-cymru.org/Services/Bogons/.

[173] Team Cymru, "Team Cymru Community Services." https://www.team-cymru.org/.

[174] The Apache Software Foundation, "Apache Hadoop." http://hadoop.apache.org/.

[175] Thonnard, O., Mees, W., and Dacier, M., "Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making," in *SIGKDD*, 2009.

[176] Triumfant, Inc., "Detecting and Remediating Malicious Attacks," tech. rep. http://www.triumfant.com/pdfs/White_Paper_Malware_Detection_Remediation_v20.pdf.

[177] U.S. Attorney's Office - Southern District of New York, "Manhattan U.S. Attorney Charges Seven Individuals for Engineering Sophisticated Internet Fraud Scheme," 2011. http://www.fbi.gov/newyork/press-releases/2011/manhattan-u.s.-attorney-charges-seven-individuals-for-engineering-sophisticated-internet-f

[178] Van Dyke Parunak, H., Nickels, A., and Frederiksen, R., "An agent-based framework for dynamical understanding of dns events (dude)," in *Proceedings of the 1st International Workshop on Agents and CyberSecurity*, ACySE '14, (New York, NY, USA), pp. 6:1–6:8, ACM, 2014.

[179] Vinay Pidathala, Darien Kindlund, Thoufique Haq and Zheng Bu, "Operation Beebus." `http://www.fireeye.com/blog/technical/cyber-exploits/2013/02/operation-beebus.html`.

[180] VirusTotal, "Private API version 2.0." `https://www.virustotal.com/en/documentation/private-api/`.

[181] VirusTotal, "VirusTotal Intelligence." `https://www.virustotal.com/en/documentation/private-api/`.

[182] Wagener, G., State, R., Dulaunoy, A., and Engel, T., "Self adaptive high interaction honeypots driven by game theory," in *Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, Jan 2009.

[183] Weimer, F., "Passive DNS replication," in *17th Annual FIRST Conference on Computer Security Incidents*, 2005.

[184] West, D. B., *Introduction to Graph Theory (2nd Edition).* Prentice Hall, 2000.

[185] Wilhelm, J. and Chiueh, T., "A forced sampled execution approach to kernel rootkit identification," in *Proceedings of the Symposium on Recent Advances in Intrusion Detection*, Jan 2007.

[186] Wolf, J., "Technical details of srizbi's domain generation algorithm." `http://blog.fireeye.com/research/2008/11/technical-details-of-srizbis-domain-generation-algorithm.html`, 2008.

[187] Wu, Z. and Leahy, R., "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," in *IEEE transactions on pattern analysis and machine . . .* , 1993.

[188] Wyke, J., "ZeroAccess." `http://sophosnews.files.wordpress.com/2012/04/zeroaccess2.pdf`, 2012.

[189] Yegneswaran, V., Barford, P., and Paxson, V., "Using honeynets for internet situational awareness," *Proceedings of the Fourth Workshop on Hot Topics in Networks (HotNets IV)*, 2005.

[190] Zhao, Y., Xie, Y., Yu, F., Ke, Q., Yu, Y., Chen, Y., and Gillum, E., "BotGraph: large scale spamming botnet detection," in *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, 2009.