UNIVERSITY OF CALIFORNIA SAN DIEGO

An Empirical Analysis on Threat Intelligence: Data Characteristics and Real-World Uses

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Guo Li

Committee in charge:

Professor Kirill Levchenko, Co-Chair
Professor Stefan Savage, Co-Chair
Professor Deian Stefan
Professor Geoffrey M. Voelker
Professor Xinyu Zhang

2020

The Dissertation of Guo Li is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Co-Chair

Co-Chair

University of California San Diego

2020

DEDICATION

To everyone who has supported me.

EPIGRAPH

"The cowards never started, and the weak died along the way. That leaves us."

Phil Knight

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisors, professor Stefan Savage and professor Kirill Levchenko. When I first joined the PhD program, I was an introvert student with little understanding of research. I was too scared to try something new or to think boldly. Thanks to my advisors, who both have very clear visions and are not afraid to take risks, I had a very exciting PhD experience. We tried a new idea for malware analysis, reverse-engineered Volkswagen engine firmware to try to disclose its emission cheating logic, and eventually moved to Threat Intelligence and try to understand the landscape of threat on the Internet. Some projects worked out, some did not, but I learned so much from this journey. They taught me to always follow my interests, not the trend, when choosing projects, and never be afraid of failure. This attitude towards research had changed me a lot, both in the way of doing research, and in the way of making life decisions. Besides, I really appreciate that both of my advisors gave me great freedom during my study, which makes my PhD a lot less stressful, and at the same time taught me to be independent very early on. It is very fortunate for me to have their support and guidance on this journey.

I also liked to thank professor Geoffrey M. Voelker. Although he is not my official advisor, he helped me a lot during two of my projects, both of which are covered in this dissertation. He also gave me a lot of valuable guidance during some of my lowest points. I really thank his kindness and caring. I would also like to thank Professors Deian Stefan and Xinyu Zhang for being on my doctoral committee and being available whenever I needed help.

PhD experience is fun but also challenging, and I can not make it without the help from so many people, especially from my “Foundry” lab mates (CSE 3142), where everyone is so kind and supportive. Our lab has a very free, open atmosphere, and we discuss almost anything freely here. With these discussions and exchange of ideas, I become more objective on thinking, and learned a lot on many different topics. We are also hard-working and spent many long nights together at lab. I will never forget those times when we laugh and fight together. I would also like to thank members from “Tiger” lab (CSE 3152), we hang out frequently and give support to

each other during hard times. The style of how they approach research problems—being very strict on defining the problem and validating with real-world examples—inspired me a lot.

Special thanks to Danny Huang and Tianyin Xu. These two senior students(now both become professors) had helped me a lot during my PhD and give me valuable advice. During my lowest points, I had multiple long conversations with them, and they shared a lot of their own stories and encouraged me to keep going. Their encouragement and the experience they shared is critical for my journey.

I would like than all the member of Sysnet group, including all the faculties, staffs and students. It has been a wonderful 6 years for me, and I had many interactions with many people during this process, in security lunch, syslunch, sysnet hiking, CNS meeting etc. All these activities not only enable me to learn a lot of stuff, but also make my journey much more enjoyable, and give me a strong feeling of belonging. Special thanks to Cindy Moore, who helped me numerous times setting up servers and fixing network issues.

Chapter 3, in part, is a reprint of the material as it appears in Proceedings of the Usenix Security 2019. *Reading the Tea Leaves: A Comparative Analysis of Threat Intelligence.* Vector Guo Li, Matthew Dunn, Paul Pearce, Damon McCoy, Geoffrey M. Voelker, Stefan Savage, Kirill Levchenko. The dissertation author was the primary investigator and author of this paper. I really like to thank my coauthors on this paper, without who I will never be able to finish this work. I like to especially thank Paul Pearce, who helped me a lot throughout the project. At the beginning, when I had little idea about what to do and was so unfamiliar with data analysis tools, he gave me critical hands-on assists. I am also very grateful to Alberto Dainotti and Alistair King for sharing the UCSD telescope data and helping me with the analysis, also professor Micheal Bailey for sharing the Mirai Botnet data. Besides, I like to thanks professor Aaron Schulman, who encouraged me a lot when the first submission of this paper was rejected, and helped me regain my confidence in the work.

Chapter 4, in part, is a reprint of the material as submitted to the Proceedings of the Usenix Security 2020. *Clairvoyance: Inferring Blacklist Use on the Internet* Vector Guo Li,

Gautam Akiwate, Yihui Chen, Geoffrey M. Voelker, Kirill Levchenko, Stefan Savage. The dissertation author was the primary investigator and author of this paper. I really appreciate the help from my coauthors, especially Gautam Akiwate. He helped me a lot on data collection and analysis, and I really learned a lot from the way he looks at data problems. Furthermore, he gave me crucial moral support during some critical times of the project. I will not be able to finish this work without his help. I also really appreciate the suggestions I received from kc Claffy and Alex Gantman regarding this project.

Last but not least, I want to thank my parents. My parents know little about research, but they know how to do things. They gave me lots of suggestions on my study, but I was too fool to realize how valuable those suggestions are. This journey is tough, and many times when I did not know what to do, when I lost my hope, when I thought about giving up, they supported me, encouraged me, gave me strength and hope to carry on and keep going. If the PhD journey is walking through a dark tunnel, then they are the torch in my hand that are always there, calm me down, show me the way and bring me warmth. They are the source of my strength and my motivation to work hard. It is very lucky for me to have such wise and supportive parents, and I will never forget their love and caring.

VITA

2014	Bachelor of Science, Peking University
2014–2020	Research Assistant, University of California San Diego
2017	Master of Science in Computer Science, University of California San Diego
2020	Doctor of Philosophy in Computer Science, University of California San Diego

ABSTRACT OF THE DISSERTATION

An Empirical Analysis on Threat Intelligence: Data Characteristics and Real-World Uses

by

Guo Li

Doctor of Philosophy in Computer Science

University of California San Diego, 2020

Professor Kirill Levchenko, Co-Chair
Professor Stefan Savage, Co-Chair

Threat Intelligence, both as a concept and a product, has been increasingly gaining prominence in the security industry. At a high-level, it is the "knowledge" that helps organizations understand and mitigate cyber-attacks. Most commonly, it refers to the collection of threat indicators—IP addresses, domain names, file hashes, etc. known to be associated with attacks. By compiling and distributing this information, it is believed that recipients will be able to better defend their systems from future attacks. Thus, there are now hundreds of vendors offering their threat intelligence solutions as a mix of public and commercial products.

However, our understanding of this data, its characterization, and the extent to which it

can meaningfully support its intended uses, is still quite limited. Furthermore, how the data is being used by organizations, how popular it is, and what impact it could have on the Internet are also not clear to our community. We lack an empirical assessment of real-world threat intelligence, both in terms of the data itself and its usage, and it is important to first understand the current status of threat intelligence, then can we reasonably discuss how to make improvements.

In this dissertation, I take an empirical approach to study threat intelligence and try to address these gaps. In particular, I explore this topic from two perspectives: 1) Studying the characteristics of threat intelligence data itself and 2) Exploring how they are used in the real-world. In particular, I formally defined a set of metrics for analyzing threat intelligence data feeds and use these measures to systematically evaluate a broad range of public and commercial feeds. Further, I ground my quantitative assessments using external measurements to investigate issues of coverage and accuracy. Finally, I designed a method using the IP ID side channel to test if a remote host is blocking traffic from a given IP address. Using this technique, I measured over 220K U.S. hosts and tested whether they consistently block connections with IPs identified on popular IP blacklists. Beyond these blacklists, I also demonstrate the evidence for more widespread use of security related traffic blocking. Together, my work provides an in-depth look into the current picture of threat intelligence and augments the knowledge of our community on this topic.

Chapter 1

Introduction

Computer security is an inherently adversarial discipline in which each "side" seeks to exploit the assumptions and limitations of the other. Attackers rely on exploiting knowledge of vulnerabilities, configuration errors or operational lapses in order to penetrate targeted systems, while defenders in turn seek to improve their resistance to such attacks by better understanding the nature of contemporary threats and the technical fingerprints left by attacker's craft. Invariably, this means that attackers are driven to innovate and diversify while defenders, in response, must continually monitor for such changes and update their operational security practices accordingly. This dynamic is present in virtually every aspect of the operational security landscape, from anti-virus signatures to the configuration of firewalls and intrusion detection systems to incident response and triage. Common to all such reifications, however, is the process of monitoring for new data on attacker behavior and using that data to update defenses and security practices. Indeed, the extent to which a defender is able to gather and analyze such data effectively defines a de facto window of vulnerability—the time during which an organization is less effective in addressing attacks due to ignorance of current attacker behaviors.

This abstract problem has given rise to a concrete demand for contemporary threat data sources that are frequently collectively referred to as *threat intelligence*. Threat intelligence is the *knowledge* that allows organizations to understand and mitigate cyber-attacks. This "knowledge" involves a wide variety of things. It can be vulnerability reports, where system and network

administrators can learn the vulnerabilities and the potential impact on their systems. It can also be IP or domain blacklists, which report the sources where attacks originate from, so people can take precautions against these indicators. It can even be an online discussion thread in an underground forum, so security experts can track what malicious actors are discussing. All of this knowledge captures contemporary information about potential threats, and therefore helps organizations better defend against them.

By far the most common form of threat intelligence are so-called *indicators of compromise:* simple observable behaviors that signal that a host or network may be compromised. These indicators are in general straightforward forensic data that are directly associated with attacks. The most notable examples are:

- **IP Addresses**: IPs known to launch certain attacks, including port scanning, vulnerability probing, brute-force login, etc.
- **Domains**: Domains known to host malware Command-and-Control servers or sending spam emails, etc.
- **URLs**: Compromised websites or phish URLs, etc.
- **File Hashes**: Indicating a file or executable known to be associated with a particular variety of malware, etc.

The presence of such indicators in a system or network is a symptom that alerts an organization to a problem. For example, if one machine in an organization communicated with a domain known to be associated with malware Command-and-Control servers, it is a strong indication that this machine is probably infected with the corresponding malware. Part of an organization's defenses should reasonably include monitoring its assets for such indicators to detect and mitigate potential compromises as they occur. And these indicators are simple enough that they can be easily integrated into defense or monitoring systems, like network firewalls or malware scanners.

While each organization naturally collects a certain amount of threat intelligence data

on its own (e.g., the attacks they repel, the e-mail spam they filter, etc.), any single entity has a limited footprint and few are instrumented to carefully segregate crisp signals of attacks from the range of ambiguity found in normal production network and system logs. Thus, it is now commonly accepted that threat intelligence data procurement is a specialized activity whereby third-party firms, and/or collections of public groups, employ a range of monitoring techniques to aggregate, filter and curate quality information about current threats. Indeed, the promised operational value of threat intelligence has created a thriving (multi-billion dollar) market [125].

Most established security firms, such as Cisco Security [28], Palo Alto Networks [92], Fortinet [51], and many specialized companies, including CrowdStrike [35], Anomali ThreatStream [10], Recorded Future [102], are all offering threat intelligence solutions. Public threat intelligence providers like Spamhaus, Abuse.ch are also getting more and more attention. The global threat intelligence market is predicated to surpass $13 Billion in 2025 [126]. With the industry thriving, there is also a rapid increase in the related research works [127], covering topics from data characterizing, effectiveness evaluation to designing better sharing systems.

From a high level, there are two major aspects of threat intelligence: *Data* and *Usage*. *Data* represents the content of threat intelligence—the actual information provided in different threat intelligence products. *Usage*, on the other hand, represents different ways people can use threat intelligence to help. Therefore, all research surrounding threat intelligence can be categorized into these two areas: research related to threat intelligence data itself, and studies on different ways to use the data. They are the two sides of the same coin.

When looking at these two general aspects, one can further take two different research approaches: *Empirical Study* and *Algorithm Exploration*. Empirical study focuses on understanding the current threat intelligence: analyzing patterns in existing data, measuring different use cases, discover shortcomings in current data format design, etc. This approach emphasizes measuring existing solutions, uncovering patterns and limitations, so the community can gain valuable insights. Algorithm exploration, on the other hand, focuses on designing new algorithms and new tools to improve current solutions, such as new threat hunting algorithms to improve threat

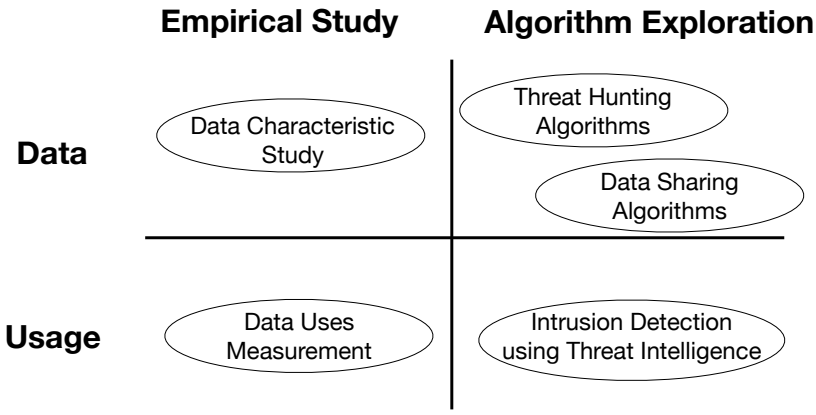


Figure 1.1. Threat Intelligence research overview and example research topics in each direction.

intelligence data quality, or better ways to utilize these data during operation. This approach emphasizes designing new tools, techniques and approaches to help the community better generate, share, and use threat intelligence data.

Therefore, threat intelligence research can be divided into four different directions (as illustrated in Figure 1.1). These are:

- Empirical analysis of threat intelligence data:

  Characterizing and understanding threat intelligence data, profiling the systems and techniques used to generate such data, measuring different threat sharing strategies and their effectiveness, etc.

- Algorithm exploration related to threat intelligence data:

  Designing algorithms for threat hunting (threat intelligence generation), defining specification for data description and protocols for data sharing, etc.

- Empirical analysis of threat intelligence usage:

  Measuring how organizations use threat intelligence, and understanding the impact such uses on the Internet, etc.

- Algorithm exploration related to threat intelligence usage:

  Designing better methods to use threat intelligence during system and network defense (e.g. increasing coverage, reducing false positives), exploring new ways to use threat intelligence

data, such as for machine learning training data, etc.

There have been many previous research works covering various topics in these four directions. Most of the works focus on threat intelligence generation techniques (all threat detection techniques, such as malware detection, phish detection, causality analysis.) and threat sharing methods (data formats, sharing protocols and sharing platforms, etc.). However, few works have taken the empirical approach and systematically evaluate the existing threat intelligence, both in terms of the data itself and its usage. Our community lacks a concrete understanding of threat intelligence in the real-world

There are two major challenges when trying to address this problem. First, organizations in general do not disclose their security settings, because these settings are considered sensitive. As a result, it is very difficult to learn what threat intelligence products people are using, and how these products perform in real systems. Second, it is very difficult, and sometimes even impossible, to get the "ground-truth" of underlying threat. Without ground-truth, one can not precisely measure threat intelligence features such as false positive rates.

In this dissertation, I will take the empirical approach and explore both the *Data* and *Usage* aspects of the threat intelligence domain. I will demonstrate in my studies that, although there is no clear way to precisely measure some features during threat intelligence study, one can give a close approximation by: *i)* carefully designing a set of evaluation metrics and calculation methods, *ii)* associating threat intelligence data with external data sources, and *iii)* utilizing indirect inferring techniques.

In exploring threat intelligence data (described in Chapter 3), I conduct a comparative analysis on existing threat intelligence products. In particular, I design formal metrics for evaluating threat intelligence data and I used these tools to evaluate 47 distinct IP address IoC feeds and 8 distinct malware IoC feeds. Through this chapter, I reveal the limitation of existing threat intelligence data and discuss potential improvements based on my findings. In exploring threat intelligence operations (described in Chapter 4), I measure how threat intelligence data is

being used on a large scale. I design a method using IP ID side channel to infer the connectivity between two Internet hosts from a third point. Using this method, I conduct a large scale Internet measurement over 220K U.S. hosts and establish their use of 9 popular public IP blacklists. I further investigate a broader use of blacklists among the hosts, and discovered over 73K hosts have shown blacklist related blocking behavior. Together, my work provides an in-depth look into the threat intelligence in the real-world and augments the knowledge of our community on this topic.

Chapter 2

Background

In this chapter, I will go through background knowledge and related work in all four research directions defined in Chapter 1. More specifically, I first introduce several threat intelligence generation techniques in section 2.1, covering different algorithms researchers have proposed for each threat category. I then discuss threat intelligence sharing and its core challenges in Section 2.2. The research works discussed in these two sections belong to the algorithmic exploration on threat intelligence data. In Section 2.3, I discuss previous measurement studies on existing threat intelligence feeds. These works fall into the direction of empirical analysis on the data, although they are severely limited in terms of scale and lack formal methodologies. Finally, I talk about the usage of threat intelligence in Section 2.4, and also go through previous algorithmic exploration and empirical surveys on the data usage.

2.1 Threat Intelligence Collection

Threat Intelligence Collection is the process of generating threat intelligence data. Cyber-attacks are happening everyday on the Internet. With vantage points, data providers can effectively capture these attacks and generate indicators of compromise. These vantage points can be specially deployed infrastructure, including honeypots and Internet telescopes; Data providers can also directly collect data from from end-users, such as anti-virus vendors collecting potential malicious executable from their customers.

However, as one can imagine, the raw data usually contains a lot of noise and can hardly serve as meaningful "intelligence". Threat intelligence collection aims to use domain knowledge and specialized algorithms to detect threats and identify attackers from the raw data, providing real valuable information about different attacks. By definition, threat detection techniques on all threat categories can be used during threat intelligence collecting. In this chapter, I will go through previous research works on several representative threat categories and discuss the techniques to detect these threats. These works fall into the algorithmic direction on threat intelligence data, base on the definition in Chapter 1.

2.1.1 Intrusion Detection

One direct way to capture threat information is to identify the attacker when a system in use is under attack, so we can capture the attacker in action. These systems can be specially deployed systems just for luring attackers, such as honeypots. They can also be real systems with real users, where people deploy detection systems and capture attacks when they are happening. The technique we use here is also the technique to protect the system in the first place: *Intrusion Detection*.

Intrusion detection aims to detect attacks in networks or systems on the fly. The techniques involved can generally be classified into two categories: misuse-based detection and anomaly-based detection. Misuse-based approaches utilize pre-defined patterns and signatures of malicious behaviors, and dynamically compare the behavior of the system against these patterns to spot potential intrusion. On the other hand, anomaly-based detection construct a model for the normal behavior of a system, then check if the current behavior is deviating from the "normal" behavior.

The performance of misuse-based detection methods depends on the quality of the pre-defined attack patterns (or detect policies), and these patterns are usually provided manually by security experts. Since different attack vectors vary a lot on their approaches and system component they touch, the patterns provided by the detection system must be able to cover a diverse set of behavior, also to be extendable, as new attacking methods are keep showing

up. Therefore, early work on this technique, such as Bro [96], Snort [105], P-BEST [78], and STAT [130] all emphasize on providing a powerful threat modeling language, which can express a broad set of threats, also easy to program to include new patterns. Recent work by Bugiel *et al.* [20] moves to new system environments (e.g. mobile system like Android), but still focuses on providing an expressive modeling language to build detect policies. These misuse-based detection methods generally have high accuracy, since the pre-defined attack patterns are usually well defined by security experts. The main disadvantage of this technique is that it can only detect modeled attacks. For new types of attacks where there are no patterns written, a misuse-based system will miss them completely.

As complementary, anomaly-based detection methods try to detect if the behavior of an application or system is different from its benign behaviors. This technique relies on modeling the normal behavior of a system in question, and since there are so many possible things a program can do, we need to simplify the "behavior" to precisely reason about it. Forrest *et al.* [50] first proposed to use the syscall sequence of a program as its behavior, since a program can only affect the operating system with syscalls, and from the security perspective, these would be the only behaviors that matter. Follow up works [74, 134, 87] explored different data models to better distinguish abnormal sequences from benign ones, using methods including data mining, Bayesian network and neural network. Recent work from Du *et al.* [39] also tries to utilize more data sources and experiment with more advanced machine learning algorithm for the detection. The advantage of these approaches is that they can capture previously unknown attacks. However, they tend to suffer more false positives, since even a normal program can show abnormal behaviors once in a while, and it is very hard to capture these cases when we first build the "normal" behavior set for a program. We can only run the programs in question for a limited amount of time and it is hard to comprehensively cover all possible states.

2.1.2 Causality Analysis

Intrusion detection techniques described above detect threats when abnormal behavior is observed. But for many advanced attacks, especially advanced persistent threats (APTs), it is not always obvious to trace from the malicious behavior, e.g. a running malware, to the source of the attacks, e.g. a phish URL that distributes the malware. This is mainly because sophisticated attackers often take precautions to hide their traces by deleting system or application logs, they can also prolong the attacks, creating a large window from the time they break into victims' machines until the time they carry out actual malicious activities. All of these can create difficulties for administrators to diagnose the intrusion and trace the source. In the context of Threat Intelligence, it is important to uncover the source of an attack, so we can provide valuable indicators of compromise for the community to defend against the same attack in an early stage.

The analysis, which tracks causal relationships between files and processes to diagnose attack provenance, is called *Attack Causality Analysis*. It is a critical technique during threat intelligence hunting. The pioneering work in this field is done by King and Chen [67]. They first defined the event dependency graph, where the nodes represent processes and files and edges represent the events between process and files, for example, a process spawns another process, or a process read or write to a file. Given a detection point, such as a suspicious file being written to the disk, or a running process initiated a suspicious network connection, the system builds a dependency graph from this point by processing event logs, then using the timestamp of each event and their causal relationship, we can trace back to the source and identify the original intrusion point.

A simple idea as it seems, it captured the fundamental logic behind causality analysis: information flow tracking. However, this simple solution quickly runs into a problem in a complex system: *dependence explosion* [54], where there are so many processes and files involved in this dependency graph, together with a large number of inter-relations, that it is very hard to identify the real attack from the haystack. The case becomes even more true when there are long-running

programs, such as a server program. The dependency associated with these programs will grow enormously over time [72], making the dependency graph even more complicated.

Many works have tried to tackle this problem. Some heuristics have been proposed to prune the graph, for example, King *et al.* [68] utilized the fact that worms try to exploit from host to host, so the traffic from a host who already has an IDS alert is more likely associated with worm attacks. Liu *et al.* [79] use the rareness of events as a metric to prioritize searching on dependency graphs. Other works try to break the entities on the graph into smaller granularity, so we can pinpoint the causal relationship between objects. For example, Goel *et al.* [54] use separate sockets reads to partition the execution of a program into different segments, so the monitoring system can figure out which action of that program corresponds with which exact network request. Lee *et al.* [72] use the prevalence of event-loops in programs to partition the execution of the program based on each loop iteration, and then associates events with specific loop iterations. Binary taint tracking has also been utilized to provide richer semantic information, as demonstrated by Ma *et al.* [82]. This topic is still an active topic today and researcher are trying to solve this from different angles.

2.1.3 Malware Analysis

Malicious software, often called malware, is always a pressing threat on the Internet. It has been used by attackers to steal sensitive user data, control victim machines to launch spam campaigns or DDoS campaigns, or encrypt valuable data to demand ransom, etc. Symantec has reported over 200 Million new malware variants just in 2018 alone [119]. Therefore, it is crucial for threat intelligence products to cover recent malware comprehensively. Malware threat intelligence data usually comes as two forms: file hashes, like MD5, that represent malware variants themselves, and IPs or domains that host Command-and-Control servers for the malware. Both forms of data is critical for organizations, as the presence of either form indicates a strong possibility of compromise, and immediate actions need to be taken. To generate these data, security companies rely on analyzing unknown binaries, collected from the Internet or uploaded

by customers, to determine if they are malicious.

To identify if an unknown binary is malware, one straightforward yet effective way is to check if the binary is a variant of known malware. Since it is nontrivial to develop a sophisticated malware program, attackers tend to just modify existing malware to generate new unseen variants. Some typical ways include code transformation (e.g. replace "mov eax, 0" with "xor eax, eax"), obfuscation, or encrypting the original binary and stores the result as data in a new executable (with a packer program). These simple techniques enable attackers to quickly generate a large number of variants from a single malware instance, and significantly increase the overhead for security experts to analyze them.

To compare a new malware sample with existing ones, we need to define a suitable representation of malware samples, from which we can calculate the similarity between them. There are two approaches to construct this representation: *Content-based* and *Behavior-based*. The content-based approach abstracts a program based on its code content, and calculates the similarity between programs by comparing their code. Early work by M. Gheorghescu *et al.* [53] propose to break the malware program into basic blocks and compare the similarity between those blocks. Dullien *et al.* [40] extract control flow graphs from malware programs and use graph similarity as the similarity between programs. These content-based approaches rely on analyzing program code itself, so they still suffer the problem of advanced code obfuscation, which can modify the code dramatically while maintaining the same functionality. This leads to the behavior- based approach, where we extract the actual behavior of malware and use that as the signature for comparison. Lee *et al.* [73] propose to use system call sequence as the signature to classify different malware samples. Bailey *et al.* [14] use the *non-transient state changes* malware causes on a system (files written, processes created) as the behavior signature, and do the comparison based on these behaviors. Holz *et al.* [103] further developed this behavior-based method, and use the actions of malware as machine learning features, and use a supervised machine learning model to conduct the comparison and classification. Bayer *et al.* [15] used taint tracking to capture a finer granularity of malware's behavior, and use this information for more

precise identification. The behavior-based approaches usually capture the behavior of malware through dynamically executing the malware samples, so it won't be affected by malware code itself, but executing the code for every variant in question impose nontrivial overhead.

When there is no existing malware to compare with, or the program in question does not match any known malware, an analysis system will need to decide if the program is malicious just based on the behavior of the program itself. Like the intrusion detection methods described in the previous section, the logic here also relies on having "specifications" that cover potential malware behaviors, and check if the analyzed program exhibits those behaviors. One common heuristic is to check if the program makes any changes to the system registry. GateKeeper [132], for example, detects spyware by monitoring if the program registers as an OS auto-start extension, such as an NT service, a tray icon in Windows, or a Unix daemon/cron job. Other tools also check different detection points, such as VICE [21], which checks for the existence of various hooks used by rootkits. More advanced systems tend to further monitor the detailed behavior of the program. For example, Kirda *et al.* [69] try to detect a popular type of spyware that uses Internet Explorer's Browser Helper Object (BHO) and toolbar interfaces to monitor a user's browsing behavior. The system uses dynamic analysis to track if the program monitors users' actions and sends out its findings to an external entity. Panorama [137], similarly, uses dynamic taint tracking to construct the information flow of an unknown program, and then use pre-defined policies(specifications) to determine if the program is malicious or not.

2.1.4 Spam Detection

Spam email, also called unsolicited bulk email or junk mail, is Internet mail that is sent to a group of recipients who have no intention to receive it. They are particularly harmful, as these emails jam users' mailboxes, engulf important personal mail, waste network bandwidth and can even crash mail-servers. Spam emails also serve as an important way of advertising products in the underground market, such as prescription drugs, illegal porn, replica of other brands, etc. It is an old yet still popular threat on the Internet. Threat intelligence spam data contains email

address used by the spammers, domains and IP addresses where spammers' mail servers are located. Organizations, and even ISPs, tend to use these data to block the incoming mail traffic.

Email providers, ISPs or anti-spam organizations usually set up *Spamtraps* to capture spam emails. Spamtraps are the email addresses that are created not for communication, but rather to lure spam. These email addresses do not belong to any person and will not involve in any kind of communication. These addresses will not be publicly announced, so only unsolicited spammers, who tend to collect target email addresses by crawling the Internet, or go through all possible lexical combinations for email names, will hit these addresses. The Spamtrap here serve as a bait to capture spammers. People also recycle long out-of-date email addresses as Spamtrap addresses.

However, simply regarding all emails received in Spamtrap as spam emails will create a substantial amount of false positives, since legitimate senders with poor data hygiene or acquisition practices can hit the traps as well. To further distinguish spam emails and consequently identify the senders, one needs to look at the content of emails themselves. Numerous statistical algorithms have been proposed by researchers to filter spam emails from legitimate ones. At its core, spam filtering can be viewed as a text categorization task: given the full text content of an email, decide whether it is spam email or benign email. A variety of supervised machine learning techniques have been tested for spam filtering. Including the naive Bayes classifier [8, 106, 108], RIPPER rule induction algorithm [33], Support Vector Machine [38], memory-based learning [9], AdaBoost [22], and maximum entropy model [139]. These algorithms all convert email headers and body into features for the machine learning model, the "bag-of-words" approach, and different feature reduction and weight assignment strategy have been explored. These models can all achieve decent accuracy, and have been tested and deployed in the real-world.

Email providers also get help from the customers to identify spam emails. Since customers can mark an email as spam manually, having a large customer base enables the email provider to collect a large number of spam emails with high accuracy, and therefore track down the domains and IP addresses used by the spammers.

2.1.5 Phishing Detection

Phishing attacks are a type of social engineering attack, where attackers disguise as trusted entities and lure victims to click on malicious links, download attachments, or provide sensitive information like credit card numbers or online credentials. These attacks can have devastating results. For individuals, this includes unauthorized access to their online accounts, the stealing of money in their bank, or identity theft for illegal activities. Moreover, phishing is often used to gain a foothold in corporate or governmental networks as a part of a larger attack, such as an advanced persistent threat (APT) event. This could result in significant financial losses, leakage of user data, and protected technologies. Hence, phishing related data is an important part of threat intelligence. These data include phishing domains and URLs, as well as IP addresses that host these websites.

Phishing attacks are often launched through phish emails, so researchers have looked into different methods to identify these phish emails. Phish emails are quite different spam emails. Spam emails are usually easy to identify, since they just try to advertise certain goods and don't try to conceal their identities. Phish emails, however, can be hard to distinguish by everyday people, since these emails deliberately pretend they are from legitimate sources, and they use multiple tactics to lure recipients to click on the link. For example, attackers exploit HTML emails and provide an HREF where the actual link is different from what is being displayed, such as `<a href="badsite.com">paypal.com</a>`. Therefore, when analyzing if an email is a phish email, researchers use text content of the email (same as spam email detection) together with phishing-specific features, like the one listed above, to build machine learning models [48, 1, 27].

Since phishing attacks mostly require victims to click on a link, the URLs and the corresponding webpages also provide us clues from which we can determine whether it is phish related. For example, in the early days, many phish URL will have IP address in it, for example, `http://135.12.44.20/index.php`. This feature is extremely rare for legitimate websites. Another characteristic is that phish domains tend to relatively long, containing multiple segments.

For example, 9794.myonlineaccounts2.abbeynational.co.uk.syrialand.com. The idea is that a very long domain can confuse people and sometimes people only read the beginning of a domain and think it is legitimate [136]. Whittaker *et al.* from Google has experimented with multiple URL related features and implemented them in their machine learning algorithm [135]. They also used other URL metadata such as PageRank to assist the detection. Another source to look at is the phishing webpage. Phishing attacks usually try to mimic other trusted websites and make people think they are visiting the original legitimate websites. They also frequently ask people to type their credentials or provide payment information, since the goal of the attackers is to steal this information. Researchers have used the content of the webpages as the feature to further enhance their algorithms. Zhang *et al.* [140] has used the word frequency on the webpage (TF-IDF) to classify the webpages. Wardman *et al.* [133] compared the content on the target webpage with other popular webpages to see if the webpages in question are trying to mimic those legitimate ones. All these resources provide additional confidence to people when deciding if a URL is a phish URL.

2.2 Threat Intelligence Sharing

How much threat one can capture is largely depending on the scope of vantage points one has on the Internet. Organizations with a larger scope of observation, like bigger Internet telescopes or a larger number of customers, tend to capture more threats. However, no entity has the capability to monitor the entire Internet, and each individual entity is only able to monitor a tiny fraction of what is happening.

This encouraged *threat intelligence sharing*, where different entities collect threat information individually and share the data with each other, so every one will get a higher coverage of potential threat. Many threat intelligence providers are offering the platform for threat intelligence sharing, including IBM X-Force Exchange [56], AlienVault Open Threat Exchange [6], Facebook ThreatExchange [45], etc. These platforms enable companies (not necessarily security companies), organizations or individual security researchers to contribute threat intelligence they

collected. Companies are also forming alliances and exchange their intelligence data within the group, like Cyber Threat Alliance [34].

One problem during threat intelligence sharing is data specification. Since different entities could collect data in different ways and also record the data in different formats. Without a clear unified data format, the data being shared will provide little benefit to the recipients, since they will not be able to understand and utilize the data. This is a nontrivial task, since this unified data format has to indicate clearly what does the data "mean". If one entity just shares a list IPs and claims these IPs are malicious IPs, it does not help much since it is crucial for the recipient to know exactly why they are malicious, e.g, because they are massively scanning the Internet, trying to brute-force log in to SSH servers, or they serve as C2 servers for malware. Since companies have different security hygiene, they will use the data differently based on its meanings. So it is critical to specify clearly the meaning of the data during sharing. Because of this, standard threat intelligence formats have been proposed and developed, notably IODEF [59], CybOX [36] and STIX [117], that try to standardize the threat intelligence presentation and sharing.

2.3 Threat Intelligence Data Study

Previous sections cover the techniques people had proposed to generate threat intelligence data and sharing them. With these techniques in place, what does the real-world threat intelligence data look like, however, is another interesting direction to study. Although individual threat detection techniques tend to promise high accuracy and coverage on target threats, after being deployed in real-world for a long time, the final data product might be far different from what people originally expected. Empirically studying the patterns and performance of threat intelligence data is an important approach to understand the current limitations, and thus provides insights for future innovations.

Several studies have examined the effectiveness of blacklist-based threat intelligence [71, 100, 101, 111, 113]. Ramachandran *et al.* [101] showed that spam blacklists are both incomplete

(missing 35% of the source IPs of spam emails captured in two spam traps), and slow in responding (20% of the spammers remain unlisted after 30 days). Sinha *et al.* [113] further confirmed this result by showing that four major spam blacklists have very high false negative rates, and analyzed the possible causes of the low coverage. Sheng *et al.* [111] studied the effectiveness of phishing blacklists, showing the lists are slow in reacting to highly transient phishing campaigns.

Other studies have analyzed the general attributes of threat intelligence data. Pitsillidis *et al.* [98] studied spam domain feeds, showing different perspectives of spam feeds, and demonstrated that different feeds are suitable for answering different questions. Thomas *et al.* [124] constructed their own threat intelligence by aggregating the abuse traffic received from six Google services, showing a lack of intersection and correlation among these different sources.

The limitations of previous measurement works are that these studies tend to only focus on specific types of threat intelligence sources, like spam or phish blacklists, and they only evaluated one aspect of the data—the operational performance, rather than generalizing the measurement and define threat intelligence metrics that can be extended beyond the work.

Little work before had defines a general measurement methodology to examine threat intelligence across a broad set of types and categories. Metcalf *et al.* [86] collected and measured IP and domain blacklists from multiple sources, but again only focused on volume and intersection analysis. One missing piece in these works is that they did not approach the problem from the perspective of consumers of threat intelligence. After all, it is the consumers that will support this industry, and research communities should look more into their needs. This is one major motivation of my work, which will be discussed in Chapter 3.

2.4 Threat Intelligence Uses

Threat intelligence data, at a high-level, promises that by compiling up-to-date information about known threats (i.e., IP addresses, domain names, file hashes, etc.), recipients of the data will be able to better defend their systems from future attacks. Therefore, a primary

use case of threat intelligence is network defense. Intrusion detection systems or firewalls can directly put the data in the system and block the corresponding IP or DNS traffic. Popular open source projects such as Snort [114], Zeek [138] all provide these functionalities. Commercial products, including Palo Alto Network firewall [93], Fortinet firewall [52], Cisco firewall [29] also incorporate threat intelligence data in their defense systems.

Besides directly taking action, threat intelligence can also be used in security monitoring and post forensic analysis. In these cases, the system raises alarms when there are matches between network activities and threat intelligence data. Threat intelligence data usually come with confidence and severity level for each individual data item. When investigating these alarms, administrators can prioritize the investigation based on the severity level. The system that in charge of collecting and organization security alarms are called Security Information and Event Management system, or SIEM system. Popular SIEM systems include Splunk [115], Sumo Logic [118] and LogRhythm NextGen SIEM [81] etc.

In academic research, threat intelligence data is usually used as extra source data to assist their study, or to evaluate the performance of their systems or algorithms. For example, Hao *et al.* [55] explored using the patterns during domain registration to detect potential malicious domains. In the study, the authors use the Spamhaus domain blacklist and URIBL to check the accuracy of their prediction. Singh *et al.* [112] studied the prevalence of Tor exit blocking in the wild, and use public and private threat intelligence sources to see how much of Tor exit IPs are listed on these sources. These are experimental use cases researchers have explored with threat intelligence data.

One question that has not been investigated is how threat intelligence products are actually being used by organizations currently in the industry. Understanding the real-world use cases gives us ideas about the adoption of threat intelligence data, which is essential information to know in the threat intelligence ecosystem. More importantly, the usage of different products offers us insight into the potential impact they could have on the Internet. As discussed in Chapter 3, false positives are relatively common in threat intelligence feeds. If an organization is

using one threat intelligence IP feed in its firewall for IP-based traffic blocking, and there is a false positive (a benign IP address) in this feed, then all the users in that organization will be affected. From another side, if one online host is added to an IP feed mistakenly, then this host will lose access to all the organizations that use this feed as a blocking ruleset. Therefore, the actual usage of the data in industry is a crucial topic that should get more attention from the security community.

But this problem is also a very challenging problem. The primary challenge is that there are many ways people can use these data. It can be directly used to block network traffic, or just raise an alarm in their Security Information and Event Management(SIEM) systems. Some use cases do not even have a well-defined behavior that we can quantitatively measure. The actions derived from threat intelligence data can also happen at different network layers. For example, an organization can deny access on network layer; it can also deny access on application layer, like HTTP 403 Forbidden. This diverse possibility of use cases makes it hard to assess this problem as a whole.

Little work has been done to try and understand how threat intelligence data is being used on a large scale. The only work that tries to look at threat intelligence data from this perspective are industrial surveys – wherein organizations fill out questionnaires. One such survey conducted by the Ponemon Institute [80], surveyed 1,200 IT and IT security practitioners, asking if they use threat intelligence products, and if they do what tools do they use that utilize the threat intelligence data. The survey also asked their user experience. SANS Institute did a similar study [109] where they surveyed 600 participants from a diverse industry background and asked questions about their threat intelligence usage. These works are limited in terms of scale, and their results are all at a very high-level. Although they offered some useful insight, they can not provide us a concrete understanding of threat intelligence uses on a large scale. My measurement, which will be discussed in Chapter 4, is the first work that systematically looks at the problem of inferring threat intelligence uses and explore its implications.

Chapter 3

Threat Intelligence Data Study

As described in the Introduction chapter, there is a surge of threat intelligence products in the recent years, and they all have eye-catching promises like high accuracy and good coverage. However, it is unclear whether the existing products on the market can actually live up to the promises, and there has been little empirical assessment of threat intelligence data or even a consensus about what such an evaluation would entail.

A thorough data characteristics analysis is critical for the community to understand the patterns and limitations of the existing threat intelligence products. After all, one needs to understand the current situation before discussing how to improve it. Another issue is that since there is little empirical analysis of the data before, security community does not even have a set of well-defined metrics for measuring and comparing different threat intelligence products. Thus, consumers of threat intelligence products have limited means to compare offerings or to factor the cost of such products into any model of the benefit to operational security.

These problems motivate my study to try to provide a grounded, empirical footing for addressing such questions. In this chapter, I will talk about my work on data characteristics study on threat intelligence. In particular, this chapter includes the following key points:

- Introduced a set of basic *threat intelligence metrics* and describe a methodology for measuring them, notably: **Volume**, **Differential Contribution**, **Exclusive Contribution**, **Latency**, **Coverage** and **Accuracy**.

- ❖ Analyze 47 distinct IP address threat intelligence sources covering six categories of threats and 8 distinct malware file hash threat intelligence sources, and report their metrics.
- ❖ Demonstrated techniques to evaluate the accuracy and coverage of certain categories of threat intelligence sources.
- ❖ I conduct the analyses in two different time periods two years apart, and demonstrate the strong consistency between the findings.

From the analysis, I find that while a few threat intelligence data sources show significant overlap, most do not. This result is consistent with the hypothesis advanced by [124] that different kinds of monitoring infrastructure will capture different kinds of attacks, but I demonstrated it in a much broader context. This also revealed that underlying this issue are broader limitations of threat intelligence sources in terms of coverage (most indicators are unique) and accuracy (false positives may limit how such data can be used operationally). Finally, I will present a longitudinal analysis suggesting that these findings are consistent over time.

3.1 Overview

The threat intelligence data collected for this study was obtained by subscribing to and pulling from numerous public and private intelligence sources. These sources ranged from simple blacklists of bad IPs/domains and file hashes, to rich threat intelligence exchanges with well labeled and structured data. I refer each item (e.g., IP address or file hash) an *indicator* (after *indicator of compromise*, the industry term for such data items).

In this section, I enumerate the threat intelligence sources, describe each source's structure and how I collected it, and then define our measurement metrics for empirically measuring these sources. When the source of the data is public, or when I have an explicit agreement to identify the provider, I have done so. However, in other cases, the data was provided on the condition of anonymity and I restrict myself to only describe the nature of the provider, but not their identity. All of the private data providers were appraised of the nature of my research, its goals and the

methodology that I planned to employ.

3.1.1 Data Set and Collection

I use several sources of threat intelligence data for my analysis:

- **Facebook ThreatExchange (FB) [44].** This is a closed-community platform that allows hundreds of companies and organizations to share and interact with various types of labeled threat data. As part of an agreement with Facebook, I collected all its data that it shared broadly. In subsequent analyses, sources with prefix "FB" indicate a unique contributor on the Facebook ThreatExchange.

- **Paid Feed Aggregator (PA).** This is a commercial paid threat intelligence data aggregation platform. It contains data collected from over a hundred other threat intelligence sources, public or private, together with its own threat data. In subsequent analyses all data sources with prefix "PA" are from unique data sources originating from this aggregator.

- **Paid IP Reputation Service.** This commercial service provides an hourly-updated blacklist of known bad IP addresses across different attack categories.

- **Public Blacklists and Reputation Feeds.** I collected indicators from public blacklists and reputation data sources, including well-known sources such as AlienVault [5], Badips [13], Abuse.ch [2] and Packetmail [91].

Threat Intelligence indicators include different types of data, such as IP address, malicious file hash, Domain, URL, etc. In this chapter, I focus the analysis on sources that provide IP addresses and file hashes, as they are the most prevalent data types in my collection.

I collect data from all sources on an hourly basis. However, both the Facebook ThreatExchange and the Paid Feed Aggregator change their members and contributions over time, creating irregular collection periods for several of the sub-data sources. Similarly, public threat feeds had varying degrees of reliability, resulting in collection gaps. In this chapter, I use the time

window from **December 1, 2017** to **July 20, 2018** for most of the analyses, as I have the largest number of active sources during this period. I eliminated duplicates sources (e.g., sources we collected individually and also found in the Paid Aggregator) and sub-sources (a source that is a branch of another source). I further break IP sources into separate categories and treat them as individual feeds, as shown in Section 3.2. This filtering leaves us with 47 IP feeds and 8 malware file hash feeds.

The ways each threat intelligence source collects data varies, and in some cases the methodology is unknown. For example, Packetmail IPs and Paid IP Reputation collect threat data themselves via honeypots, analyzing malware, etc. Other sources, such as Badips or the Facebook ThreatExchange, collect their indicators from general users or organizations—e.g., entities may be attacked and submit the indicators to these threat intelligence services. These services then aggregate the data and report it to their subscribers. Through this level of aggregation the precise collection methodologies and data providence can be lost.

3.1.2 Data Source Structure

threat intelligence sources in my corpus structure and present data in different ways. Part of the challenge in producing cross-dataset metrics is normalizing both the structure of the data as well as its *meaning*. A major structural difference that influences my analysis occurs between data sources that provide data in *snapshots* and data sources that provide *events*.

Snapshot. Snapshot feeds provide periodic snapshots of a set of indicators. More formally, a snapshot is a set of indicators that is a function of time. It defines, for a given point in time, the set of indicators that are members of the data source. Snapshot feeds imply *state*: at any given time, there is a set of indicators that are *in* the feed. A typical snapshot source is a published list of IPs periodically updated by its maintainer. For example, a list of command-and-control IP addresses for a botnet may be published as a snapshot feed subject to periodic updates.

All feeds of file hashes are snapshots and are *monotonic* in the sense that indicators are only added, not removed, from the feed. Hashes are a proxy for the file content, which does not

change (malicious file content will not change to benign in the future).

Event. In contrast, event feeds report newly discovered indicators. More formally, an event source is a set of indicators that is a function of a time *interval.* For a given time interval, the source provides a set of indicators that were seen or discovered in that time interval. Subscribers of these feeds query data by asking for new indicators added in a recent time window. For example, a user might, once a day, request the set of indicators that appeared in the last 24 hours.

This structural difference is a major challenge when evaluating feeds comparatively. I need to normalize the difference to make a fair comparison, especially for IP feeds. From a threat intelligence consumer's perspective, an *event* feed does not indicate when an indicator will expire, so it is up to the consumer to act on the age of indicators. Put another way, the expiration dates of indicators are decided by how users query the feed: if a user asks for the indicators seen in the last 30 days when quering data, then there is an implicit 30-day valid time window for these indicators.

In this chapter, I choose a 30-day valid period for all the indicators I collected from event feeds—the same valid period used in several snapshot feeds, and also a common query window option offered by event feeds. I then convert these event feeds into snapshot feeds and evaluate all of them in a unified fashion.

3.1.3 Threat Intelligence Metrics

The aim of this work is to develop *threat intelligence metrics* that allow a threat intelligence consumer to compare threat intelligence sources and reason about their fitness for a particular purpose. To this end, I propose six concrete metrics: *Volume*, *Differential contribution*, *Exclusive contribution*, *Latency*, *Accuracy* and *Coverage*.

✧ **Volume.** I define the *volume* of a feed to be the total number of indicators appearing in a feed over the measurement interval. Volume is the simplest threat intelligence metric and has an established history in prior work [64, 71, 76, 98, 111, 113, 124]. It is also useful to study the daily *rate* of a feed, which quantifies the amount of data appearing in a feed on a daily basis.

Rationale: To a first approximation, volume captures how much information a feed provides to the consumer. For a feed without false positives (see *accuracy* below), and if every indicator has equal value to the consumer, one would prefer a feed of greater volume to a feed of lesser volume. Of course, indicators do not all have the same value to consumers: knowing the IP address of a host probing the entire Internet for decades-old vulnerabilities is less useful than the address of a scanner targeting organizations in your sector looking to exploit zero-day vulnerabilities.

✧ **Differential contribution.** The *differential contribution* of one feed with respect to another is the number of indicators in the first that are not in the second during the same measurement period. We define differential contribution relative to the size of the first feed, so that the differential contribution of feed A with respect to feed B is $\text{Diff}_{A,B} = |A \setminus B|/|A|$. Thus, $\text{Diff}_{A,B} = 1$ indicates that the two feeds have no elements in common, and $\text{Diff}_{A,B} = 0$ indicates that every indicator in A also appears in B. It is sometimes useful to consider the complement of differential contribution, namely the normalized *intersection* of A in B, given by $\text{Int}_{A,B} = |A \cap B|/|A| = 1 - \text{Diff}_{A,B}$.

Rationale: For a consumer, it is often useful to know how many *additional* indicators a feed offers relative to one or more feeds that the consumer has already. Thus, if a consumer already has feed A and is considering paying for feed B, then $\text{Diff}_{A,B}$ indicates how many new indicators feed A will provide.

✧ **Exclusive contribution.** The *exclusive contribution* of a feed with respect to a set of other feeds is the proportion of indicators unique to a feed, that is, the proportion of indicators that occur in the feed but no others. Formally, the exclusive contribution of feed A is defined as $\text{Uniq}_{A,B} = |A \setminus \bigcup_{B \neq A} B|/|A|$. Thus, $\text{Uniq}_{A,B} = 0$ means that every element of feed A appears in some other feeds, while $\text{Uniq}_{A,B} = 1$ means no element of A appears in any other feed.

Rationale: Like differential contribution, exclusive contribution tells a threat intelligence consumer how much of a feed is different. However, exclusive contribution compares a feed to all other feeds available for comparison, while differential contribution compares a feed to

just another feed. From a threat intelligence consumer's perspective, exclusive contribution is a general measure of a feed's unique value.

✧ **Latency.** For an indicator that occurs in two or more feeds, its *latency* in a feed is the elapsed time between its first appearance in any feed and its appearance in the feed in question. In the feed where an indicator first appeared, its latency is zero. For all other feeds, the latency indicates how much later the same indicators appears in those feeds. Taster's Choice [98] referred to latency as *relative first appearance time*. (I find the term *latency* to be more succinct without loss of clarity.) Since latency is defined for one indicator, for a feed it makes sense to consider statistics of the distribution of indicator latencies, such as the median indicator latency.

Rationale: Latency characterizes how quickly a feed includes new threats: the sooner a feed includes a threat, the more effective it is at helping consumers protect their systems. Indeed, several studies report on the impact of feed latency on its effectiveness at thwarting spam [26, 101].

The metrics above are defined without regard for the *meaning* of the indicators in a feed. One can calculate the volume of a single feed or the differential contribution of one feed with respect to another regardless of what the feed purports to contain. While these metrics are easy to compute, they do little to tell us about the fitness of a feed for a particular purpose. For this, I need to consider the meaning or purpose of the feed data, as advertised by the feed provider. I define the following two metrics.

✧ **Accuracy.** The *accuracy* of a feed is the proportion of indicators in a feed that are correctly included in the feed. Feed accuracy is analogous to *precision* in Information Retrieval. This metric presumes that the description of the feed is well-defined and describes a set of elements that should be in the feed given perfect knowledge. In practice, researchers have neither perfect knowledge nor a perfect description of what a feed should contain. In some cases, however, I can construct a set A^- of elements that should definitely not be in a feed A. Then $\mathrm{Acc}_A \leq |A \setminus A^-|/|A|$.

Rationale: The accuracy metric tells a threat intelligence consumer how many false

positives to expect when using a feed, and, therefore, dictates how a feed can be used. For example, if a consumer automatically blocks all traffic to IP addresses appearing in a feed, then false positives may cause disruption in an enterprise by blocking traffic to legitimate sites. On the other hand, consumers may tolerate some false positives if a feed is only used to gain additional insight during an investigation.

✧ **Coverage.** The *coverage* of a feed is the proportion of the intended indicators contained in a feed. Feed coverage is analogous to *recall* in Information Retrieval. Like accuracy, coverage presumes that the description of the feed is sufficient to determine which elements should be in a feed, given perfect knowledge. In some cases, it is possible to construct a set A^+ of elements that should be in a feed. I can then upper-bound the coverage $\text{Cov}_A \leq |A|/|A^+|$.

Rationale: For a feed consumer who aims to obtain complete protection from a specific kind of threat, coverage is a measure of how much protection a feed will provide. For example, an organization that wants to protect itself from a particular botnet will want to maximize its coverage of that botnet's command-and-control servers or infection vectors.

In the following two sections, I use these metrics to evaluate two types of threat intelligence: IP address feeds and file hash feeds.

3.2 IP Threat Intelligence

One of the most common forms of threat intelligence are feeds of IP addresses considered malicious, suspicious, or otherwise untrustworthy. This type of threat intelligence dates back at least to the early spam and intrusion detection blacklists, many of which are still active today such as SpamhausSBL [122], CBL [23] and SORBS [121]. Here, I apply the metrics described above to quantify the differences between 47 different IP address threat intelligence feeds.

3.2.1 Feed Categorization

IP address threat intelligence feeds have different meanings, and, therefore, purposes. To meaningfully compare feeds to each other, I first group feeds into *categories* of feeds whose

indicators have the same intended meaning. Unfortunately, there is no standard or widely accepted taxonomy of IP threat intelligence feeds. To group feeds into semantic categories, I use metadata associated with the feed as well as descriptions of the feed provided by the producer, as described below.

Metadata. Some feeds provide category information with each indicator as metadata. More specifically, all of the Paid Aggregator feeds, Alienvault IP Reputation and Paid IP Reputation include this category metadata. In this case, I use its pre-assigned category in the feed. Facebook ThreatExchange feeds do not include category information in the metadata, but instead provide a descriptive phrase with each indicator. I then derive its category based on the description.

Feed description. For feeds without metadata, I rely on online descriptions of each feed, where available, to determine its semantic category. For example, the website of feed Nothink SSH [89] describes that the feed reports brute-force login attempts on its corresponding honeypot, which indicates the feed belongs to brute-force category.

I grouped the IP feeds into categories derived from the information above. In this work, I analyze six of the most prominent categories:

- **Scan:** Hosts doing port or vulnerability scans.
- **Brute-force:** Hosts making brute force login attempts.
- **Malware:** Malware C&C and distribution servers.
- **Exploit:** Hosts trying to remotely exploit vulnerabilities.
- **Botnet:** Compromised hosts belonging to a botnet.
- **Spam:** Hosts that sent spam or should not originate email.

Table 3.1 lists the feeds, grouped by category, used in the rest of this section. The symbols ○ and △ before the feed name indicate whether the feed is a snapshot feed or an event feed, respectively (see Section 3.1.2). All data was collected during my measurement period, **December 1st, 2017** to **July 20th, 2018**. Note that a few feeds, like Paid IP Reputation, appear in multiple categories.

In these feeds, indicators are associated with different categories via attached metadata. I split these feeds into multiple virtual feeds each containing indicators belonging to the same category.

3.2.2 Volume

Volume is one of the oldest and simplest threat intelligence metrics representing how informative each data source is. Table 3.1 and Table 3.2 shows the total number of unique IP addresses collected from each feed during the measurement period, under column *Volume*. A ○ denotes a *snapshot feed* and △ indicates an *event feed* (Section 3.1.2). *Volume* is the total number of IPs collected during my measurement period. *Exclusive* is the exclusive contribution of each feed (Section 3.2.4). *Avg. Rate* is the number of average daily new IPs added in the feed (Section 3.2.6), and *Avg. Size* is the average working set size of each feed (Section 3.2.2). Feeds are listed in order of decreasing volume, grouped by category. The numbers I show are after the removal of invalid entries identified by the sources themselves. Column *Avg. Rate* shows the average number of new IPs I received per day, and *Avg. Size* lists the average daily working set size of each feed, that is, the average size of the snapshot.

✦ **Finding:** Feeds vary dramatically in volume. Within every category, big feeds can contain orders of magnitude more data than small feeds. For example, in the scan category, I saw over 361,004 unique IP addresses in DShield IPs but only 1,572 unique addresses in PA Analyst in the same time period. Clearly, volume is a major differentiator for feeds.

Average daily rate represents the amount of new indicators collected from a feed each day. Some feeds may have large volume but low daily rates, like Feodo IP Blacklist in the malware category. This means most indicators one can get from that feed are old data present in the feed before our measurement started. On the other hand, the average rate of a feed could be greater than the volume would suggest, like Nothink SSH in the brute-force category. This is due to the fact that indicators can be added and removed multiple times in a feed. In general, IP indicators tend to be added in a feed only once: 37 among 47 IP feeds have over 80% of their indicators appearing only once, and 30 of them have this rate over 90%. One reason is that some snapshot

Table 3.1. IP threat intelligence feeds used in the study (Part I)

Feed	*Volume*	*Exclusive*	*Avg. Rate*	*Avg. Size*
Scan Feeds				
○ PA AlienVault IPs[1]	425,967	48.6%	1,359	128,821
△ DShield IPs	361,004	31.1%	1,556	69,526
○ PA Packetmail ramnode	258,719	62.0%	870	78,974
△ Packetmail IPs	246,920	48.6%	942	29,751
○ Paid IP Reputation	204,491	75.6%	1,362	8,756
○ PA Lab Scan	169,078	63.1%	869	9,775
○ PA Snort BlockList	19,085	96.3%	56	4,000
△ FB Aggregator_1	6,066	71.3%	24	693
○ PA Analyst	1,572	34.5%	6.3	462
Botnet Feeds				
○ PA Analyst	180,034	99.0%	697	54,800
○ PA CI Army	103,281	97.1%	332	30,388
○ Paid IP Reputation	77,600	99.9%	567	4,278
○ PA Botscout IPs	23,805	93.8%	81	7,180
○ PA VoIP Blacklist	10,712	88.0%	40	3,633
○ PA Compromised IPs	7,679	87.0%	21	2,392
○ PA Blocklist Bots	4,179	80.7%	16	1,160
○ PA Project Honeypot	2,600	86.5%	8.5	812
Brute-force Feeds				
△ Badips SSH	542,167	84.1%	2,379	86,677
△ Badips Badbots	91,553	70.8%	559	17,577
○ Paid IP Reputation	89,671	52.8%	483	3,705
○ PA Brute-Force	41,394	92.1%	138	14,540
△ Badips Username Notfound	37,198	54.2%	179	3662.8
△ Haley SSH	31,115	43.6%	40	1,224
△ FB Aggregator_2	22,398	77.3%	74	2,086
△ Nothink SSH	20,325	62.7%	224	12,577
△ Dangerrulez Brute	10,142	4.88%	37	1,102
Malware Feeds				
○ Paid IP Reputation	234,470	99.1%	1,113	22,569
△ FB Malicious IPs	30,728	99.9%	129	3,873
○ Feodo IP Blacklist	1,440	47.7%	1.3	1,159
○ PA Lab Malware	1,184	84.6%	3.5	366
△ Malc0de IP Blacklist	865	61.0%	2.9	86.6
○ PA Bambenek C2 IPs	785	92.1%	3.4	97.9
○ PA SSL Malware IPs	676	53.9%	2.9	84.0
○ PA Analyst	492	79.8%	2.1	149
○ PA Abuse.ch Ransomware	256	7.03%	1.6	117
○ PA Mal-Traffic-Anal	251	60.5%	0.9	72
○ Zeus IP Blacklist	185	49.1%	0.5	101

Table 3.2. IP threat intelligence feeds used in the study (Part II)

Feed	*Volume*	*Exclusive*	*Avg. Rate*	*Avg. Size*
Exploit Feeds				
△ Badips HTTP	305,020	97.6%	1,592	22,644
△ Badips FTP	285,329	97.5%	1,313	27,601
△ Badips DNS	46,813	99.3%	231	4,758
△ Badips RFI	3,642	91.4%	16	104
△ Badips SQL	737	79.5%	4.4	99.2
Spam Feeds				
○ Paid IP Reputation	543,583	99.9%	3,280	6,551
△ Badips Postfix	328,258	90.5%	842	27,951
△ Badips Spam	302,105	89.3%	1,454	30,197
○ PA Botscout IPs	14,514	89.3%	49	4,390
○ Alienvault IP Reputation	11,292	96.6%	48	1,328

feeds maintain a valid period for each indicator, as I found in all *PA* feeds where the expiration date of each indicator is explicitly recorded. When the same indicator is discovered again by a feed before its expiration time, the feed will just extend its expiration date, so this occurrence will not be captured if I simply subtract the old data from the newly collected data to derive what is added on a day. For event feeds and snapshot feeds in *PA* where I can precisely track every occurrence of each indicator, I further examed data occurrence frequency and still found that the vast majority of IPs in feeds only occurred once—an observation that relates to the dynamics of cyber threats themselves.

Nothink SSH, as I mentioned above, is a notable exception. It has over 64% of its indicators appearing 7 times in my data set. After investigating, we found that this feed posts all its previous data at the end of every month, behavior very likely due to the feed provider instead of the underlying threats.

The working set size defines the daily average amount of indicators users need to store in their system to use a feed (the storage cost of using a feed). The average working set size is largely decided by the valid period length of the indicators, controlled either by the feed (snapshot feeds) or the user (event feeds). The longer the valid period is, the larger the working set will be. Different snapshoot feeds have different choices for this valid period: PA AlienVault IPs in the

scan category sets a 90-day valid period for every indicator added to the feed, while PA Abuse.ch Ransomware uses a 30-day period. Although one would not know the data expiration mechanism used by snapshot feeds other than *PA* feeds, as there is no related information recorded, I can still roughly estimate this by checking the *durations* of their indicators—the time between an indicator being added and being removed. Four Paid IP Reputation feeds have more than 85% of durations shorter than 10 days, while the one in the malware category has more than 40% that span longer than 20 days. Feodo IP Blacklist has over 99% of its indicators valid for my entire measurement period, while over 70% of durations in the Zeus IP Blacklist are less than 6 days. I did not observe a clear pattern regarding how each snapshot feed handles the expiration of indicators.

3.2.3 Differential Contribution and Intersection

The differential contribution metric measures the number of indicators in one feed that are not in another. Equivalently, one can consider the intersection of two feeds, which is the number of elements in one feed that are present in the other, normalized by the size of the first: $|A \cap B|/|A|$. Figure 3.1 shows the intersection relationship of all feeds in the study. Each cell in the matrix represents the number of elements in both feeds, normalized by the size of the feed spanning the rows on the table. That is, A, in the expression above, ranges over rows, and B over columns of the matrix. Darker (more saturated) colors indicate greater intersection. Comparisons of feeds within a category are shaded red and comparisons of feeds between different categories are shaded blue. Note that the matrix is asymmetric, because, in general, $|A \cap B|/|A| \neq |A \cap B|/|B|$. Elements of the matrix are in the same order as in Table 3.1.

✦ **Finding:** Feeds in scan and brute-force categories have higher pairwise intersections: Half of the pairwise intersection rates in two categories are greater than 5%. The scan category has 29 out of 72 pairs (excluding self comparisons) with an intersection rate larger than 10%, and the same case occurred in 19 out of 72 pairs in the brute-force category.

On the other side, feeds in the botnet, exploit, malware and spam category do not share

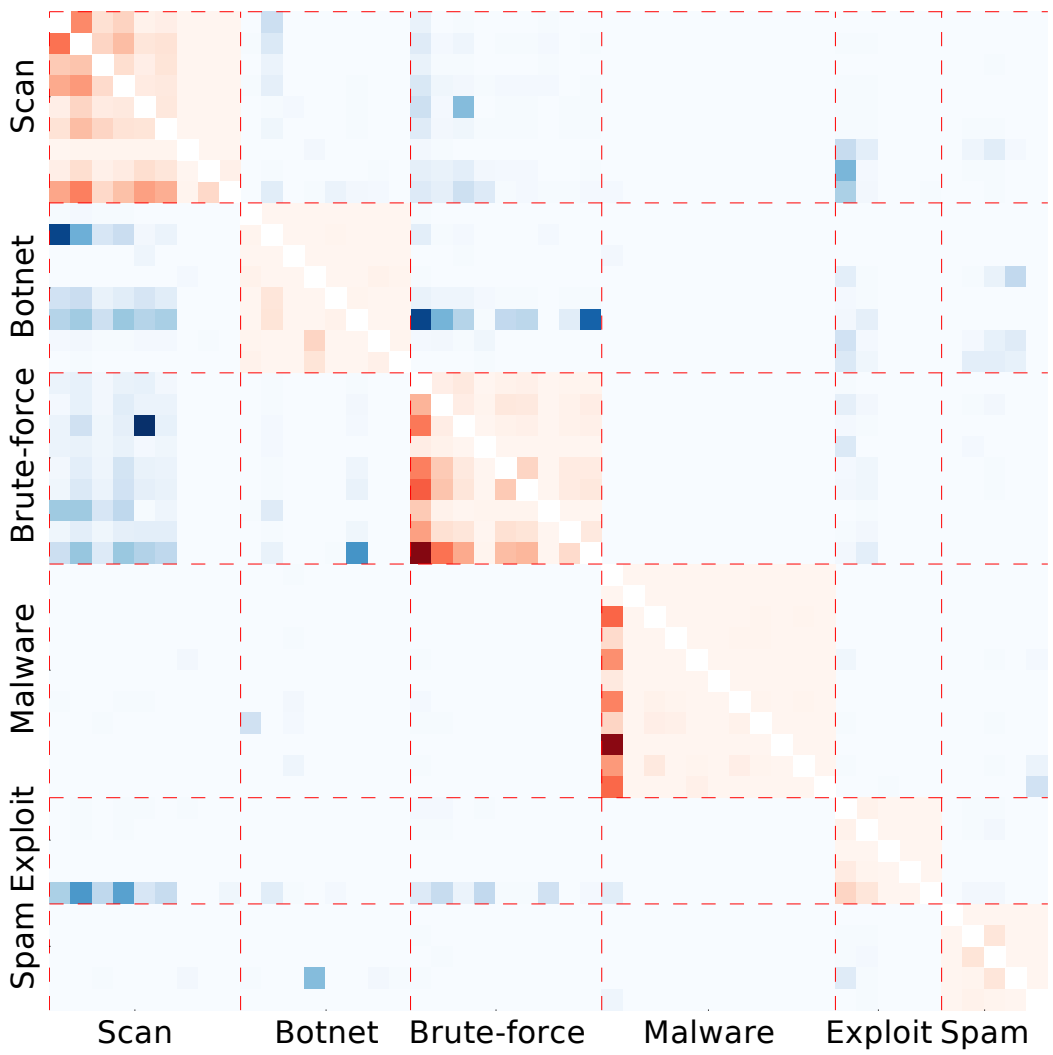


Figure 3.1. Feed intersection for all IP feeds. Each row/column represents a feed, shown in the same order as Table 3.1. Darker (more saturated) colors indicate greater intersection.

much data between each other: all 4 categories have more than three-quarters of pairwise intersection rates less than 1%. A few big feeds in these categories can share a significant amount of data with some small feeds in the same category—a characteristic that appears as a dark vertical line within its category in Figure 3.1. Paid IP Reputation in the malware category, for example, shares over 30% of 6 other malware feeds. But the intersections among the vast majority of feeds in these 4 categories are low. This finding is consistent with prior work [86, 124], but I provide a more comprehensive view regarding different categories.

Figure 3.1 also shows the relation between feeds across different categories. One can clearly see a relation between scan and brute-force feeds: multiple scan feeds have non-trivial intersection with feeds in the brute-force category. In fact, 23.1% of all 760,263 brute-force IPs I collected are also included by scan feeds in my dataset. There are also three botnet feeds—PA CI Army, PA VoIP Blacklist and PA Compromised IPs—that have over 10% of its data shared with multiple feeds in the scan category.

3.2.4 Exclusive Contribution

Exclusive contribution represents the number of indicators in a feed that are in no other feeds. I calculate each feed's exclusive contribution among all the feeds in the same category, emphasizing their uniqueness regarding the scope of data they claim to report. Each feed's exclusive contribution is presented in Table 3.1 in column *Exclusive*, calculated based on its volume.

✦ **Finding:** As I already observed in Section 3.2.3, botnet, exploit and spam feeds have relatively low pairwise intersections. Consequently, the feeds in these four categories have high exclusive contribution rates in general: the median exclusive contribution rates of these four categories are 90.9%, 97.5% and 90.5%, respectively. The malware category has a low median exclusive rate, since multiple small feeds have non-trivial intersection with the largest feed Paid IP Reputation, but the two largest feeds in malware both have a exclusive rate over 99%. Scan and brute-force feeds have more intersection within its category, and their exclusive rates are lower: 62.0% median rate in scan and 62.7% in brute-force, and the top two largest feeds in both categories have an exclusive rate below 85%.

If one assumes a process where a feed is more likely to have popular elements, then smaller feeds would be subsumed by larger feeds. Yet, for some small feeds like Malc0de IP Blacklist in the malware and PA Project Honeypot in the botnet categories, even though they are several orders of magnitude smaller than the largest feeds in their categories, a significant proportion of their indicators is still unique to the feed. When I aggregate the data in each

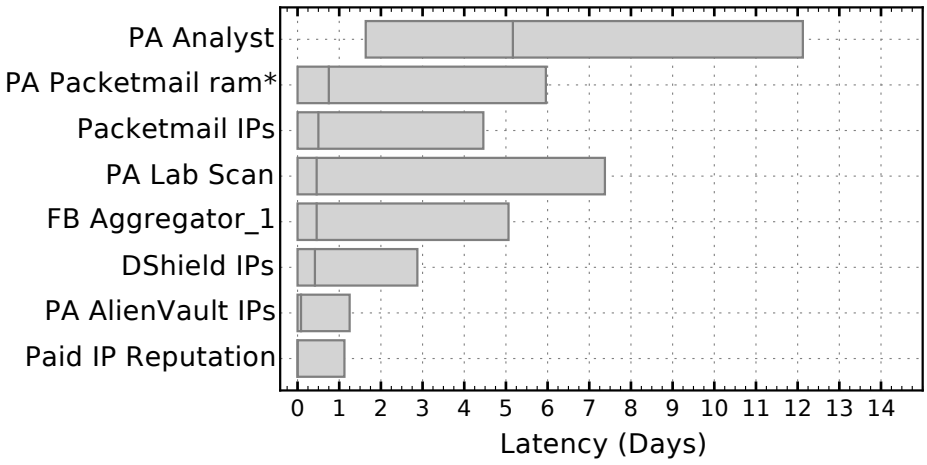
category, 73% of all scan feed indicators are unique to a single feed and 88% of brute force feed indicators are unique to one feed. For other categories, over 97% of elements in the category are unique to a single feed. This result agrees with previous work that most data in threat intelligence feeds is unique [86, 124].

3.2.5 Latency

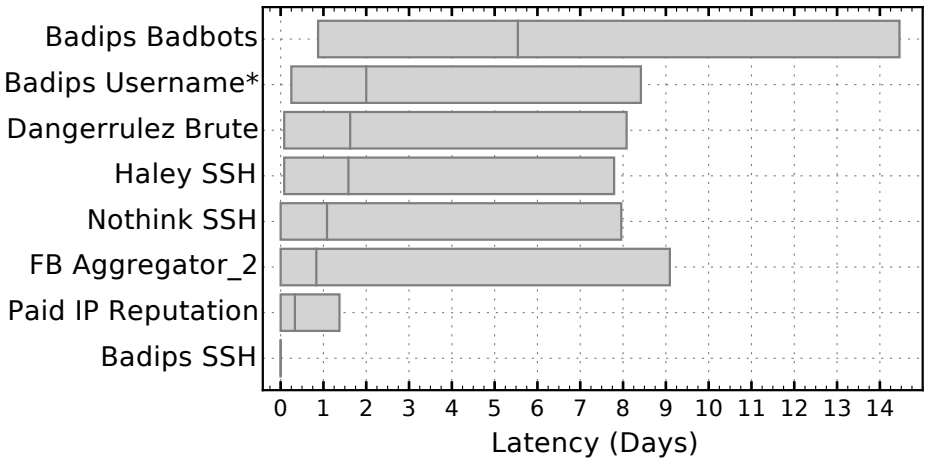
Feed latency measures how quickly a feed reports new threat indicators. The sooner a feed can report potential threats, the more valuable it is for consumers. The absolute latency of an indicator in a feed is the time from the beginning of the corresponding event until when the indicator shows up in the feed. However, it is difficult to know the actual time when an event begins from the threat intelligence data. Instead, I measure the *relative latency*, which is the delay of an indicator in one feed to be the time between its appearance in that feed and the first seen among all the feeds.

Relative latency can only be calculated for indicators that occur in at least two feeds. As discussed in Section 3.2.4, the number of common indicators in the botnet, malware, exploit and spam feeds is very low (fewer than 3% of elements occur in more than one feed). Relative latency calculated for these feeds is less meaningful. For this analysis, therefore, I focus on scan and brute-force feeds.

Another issue is the time sensitivity of IP threats. An event that originated from an IP address, like scanning activity or a brute-force attack, will not last forever. If one scan feed reports an IP address today and another feed reports the same IP three months later, it would make little sense to consider them as one scanning event and label the second occurrence as being three months late. Unfortunately, there is no easy way I can clearly distinguish events from each other. Here I use a one-month window to restrict an event, assuming that the same attack from one source will not last for more than 30 days; although arbitrary, it provides a reasonably conservative threshold, and experimenting with other thresholds produced similar overall results. More specifically, I calculate relative latency by tracking the first occurrence of IPs in all feeds



(a) Latency distribution in scan feeds



(b) Latency distribution in brute-force feeds

Figure 3.2. Distribution of indicators' latency in scan and brute-force feeds.

in a category, then recording the latency of the following occurrences while excluding ones that occur after 30 days. By just using the first appearance of each IP as the base, I avoid the uncertainty caused by multiple occurrence of indicators and different valid periods used among feeds.

Figures 3.2a and 3.2b show the relative latency distribution among feeds in the scan and brute-force categories, in hours. Each box shows the latency distribution of shared IPs in the feed calculated in hours from 25 percentile to 75 percentile, with the middle line indicating

the median. ("Badips Username*" here is the abbreviation for feed name Badips Username Notfound; "PA Packetmail Ram*" for PA Packetmail Ramnode) I focus on just those feeds that have over 10% of their data shared with others to ensure the analysis can represent the latency distribution of the overall feed. There is one feed in each category (PA Snort BlockList in scan and PA Brute-Force in brute-force) that is excluded from the figure.

✦ **Finding:** From the distribution boxes one can see that Paid IP Reputation in scan and Badips SSH in brute-force are the fastest feeds in their category, as they have the lowest median and 75th percentile latencies. On the other hand, PA Analyst in scan and Badips Badbots in brute-force are the slowest feeds. Figure 3.2a shows that all scan feeds except one have their 25th percentile latency equal to 0, indicating these feeds, across different sizes, all reported a significant portion of their shared data first. A similar case also happens in the brute-force category.

One may reasonably ask whether large feeds report data sooner than small feeds. The result shows that this is not always the case. FB Aggregator$_1$ is the second smallest feed in the scan category, yet it is no slower than several other feeds which have over 10 times of its daily rate. Badips Badbots, on the other hand, has the second largest rate in brute-force category, but it is slower than all the other feeds in the brute-force category. Feeds that are small in volume can still report a lot of their data first.

Another factor that could affect latency is whether feeds copy data from each other. For example, 93% of Dangerrulez Brute also appears in Badips SSH. If this is the case, I expect Dangerrulez Brute will be faster than Badips SSH on reporting their shared data. However, I compared the relative latency between just two feeds and found Badips SSH reported 88% of their shared indicators first. I further conducted this pairwise latency comparison between all feeds in scan, brute-force and malware (since Paid IP Reputation shares non-trivial amount of data with a few small feeds in the malware category), and did not see a clear latency advantage between any two feeds. Note that this observation does *not* prove there is no data copying, since the shared data between two feeds might partially come from copying and partially from the

feeds' own data collection. Furthermore, my latency analysis is at a one-hour granularity.

3.2.6 Accuracy

Accuracy measures the rate of false positives in a feed. A false positive is an indicator that data is labeled with a category to which it does not belong. For example, an IP address found in a scan feed that has not conducted any Internet scanning is one such false positive. As well, even if a given IP is in fact associated with malicious activity, if it is not unambiguously actionable (e.g., Google's DNS at 8.8.8.8 is used by malicious and benign software alike) then for many use cases it must also be treated as a false positive. False positives are problematic for a variety of reasons, but particularly because they can have adverse operational consequences. For example, one might reasonably desire to block all new network connections to and from IP addresses reported as hosting malicious activity (indeed, this use is one of the promises of threat intelligence). False positives in such feeds, though, could lead to blocking legitimate connections as well. Thus, the degree of accuracy for a feed may preclude certain use cases.

Unfortunately, determining which IPs belong in a feed and which do not can be extremely challenging. In fact, at any reasonable scale, I am unaware of any method for unambiguously and comprehensively establishing "ground truth" on this matter. Instead, in this section I report on a proxy for accuracy that provides a conservative assessment of this question. To wit, I assemble a *whitelist* of IP addresses that either should not reasonably be included in a feed, or that, if included, would cause significant disruption. The presence of such IPs in a feed are clearly false positives and thus define an upper bound on a feed's accuracy. I populate my list from three sources: unroutable IPs, IPs associated with top Alexa domains, and IPs of major content distribution networks (CDNs). The detail result for the accuracy analysis is presented in Table 3.3 and Table 3.4. *Unrt* is fraction of unroutable addresses in each feed (Section 3.2.6). *Alexa Top* is the number of IPs intersected with top Alexa domain IP addresses, and *CDNs* is the number of IPs intersected with top CDN provider IP addresses. I will explain more about each column in below.

Table 3.3. IP threat intelligence feeds accuracy overview (Part I)

Feed	*Added*	*Unrt*	*Alexa*	*CDNs*
Scan Feeds				
PA AlienVault IPs	313,175	0.0%	1	0
DShield IPs	339,805	0.03%	68	62
PA Packetmail ramnode	200,568	<0.01%	0	0
Packetmail IPs	211,081	0.0%	0	0
Paid IP Reputation	200,915	1.65%	6	21
PA Lab Scan	169,037	<0.01%	0	0
PA Snort BlockList	12,957	0.42%	1	0
FB Aggregator_1	5,601	0.0%	0	0
PA Analyst	1,451	0.41%	0	0
Botnet Feeds				
PA Analyst	180,034	<0.01%	0	0
PA CI Army	76,125	<0.01%	0	0
Paid IP Reputation	73,710	1.66%	6	74
PA Botscout IPs	18,638	0.09%	1	0
PA VoIP Blacklist	9,290	0.32%	0	0
PA Compromised IPs	4,883	0.0%	0	0
PA Blocklist Bots	3,594	0.0%	0	0
PA Project Honeypot	1,947	0.0%	0	0
Brute-force Feeds				
Badips SSH	456,605	0.19%	217	1
Badips Badbots	91,553	1.04%	46	1,251
Paid IP Reputation	87,524	0.03%	0	10
PA Brute-Force	31,555	0.0%	0	0
Badips Username Notfound	37,198	0.53%	4	0
Haley SSH	8,784	0.03%	0	0
FB Aggregator_2	17,779	0.0%	0	0
Nothink SSH	20,325	1.51%	2	0
Dangerrulez Brute	8,247	0.0%	0	0
Malware Feeds				
Paid IP Reputation	217,073	0.13%	291	3,489
FB Malicious IPs	29,840	2.14%	2	0
Feodo IP Blacklist	296	0.0%	0	0
PA Lab Malware	806	2.85%	0	0
Malc0de IP Blacklist	668	0.0%	8	11
PA Bambenek C2 IPs	777	9.13%	0	0
PA SSL Malware IPs	674	0.0%	0	0
PA Analyst	486	0.0%	0	0
PA Abuse.ch Ransomware	256	3.12%	0	0
PA Mal-Traffic-Anal	193	0.51%	0	0
Zeus IP Blacklist	67	0.0%	1	0

Table 3.4. IP threat intelligence feeds accuracy overview (Part II)

Feed	*Added*	*Unrt*	*Alexa*	*CDNs*
Exploit Feeds				
Badips HTTP	305,020	0.67%	16	2,590
Badips FTP	285,329	1.33%	14	2
Badips DNS	46,813	0.50%	119	244
Badips RFI	3,642	2.22%	0	0
Badips SQL	737	1.89%	0	1
Spam Feeds				
Paid IP Reputation	543,546	78.7%	1	0
Badips Spam	302,105	0.02%	19	0
Badips Postfix	193,674	1.29%	18	1
PA Botscout IPs	11,358	0.06%	0	0
Alienvault IP Reputation	10,414	0.07%	63	1,040

Unroutable IPs. Unroutable IPs are IP addresses that were not BGP-routable *when they first appeared* in a feed, as established by contemporaneous data in the RouteViews service [128]. While such IPs could have appeared in the source address field of a packet (i.e., due to address spoofing), it would not be possible to complete a TCP handshake. Feeds that imply that such an interaction took place should not include such IPs. For example, feeds in the Brute-force category imply that the IPs they contain were involved in brute-force login attempts, but this could not have taken place if the IPs are not routable. While including unroutable addresses in a feed is not, in itself, a problem, their inclusion suggests a quality control issue with the feed, casting shade on the validity of other indicators in the feed.

To allow for some delays in the feed, I check if an IP was routable at any time in the seven days prior to its first appearance in a feed, and if it had, I do not count it as unroutable. Table 3.3, column *Unrt*, shows the fraction of IP indicators that were not routable at any time in the seven days prior to appearing in the feed. This analysis is only conducted for the IPs that are added after my measurement started. The number of such IPs is shown in column *Added*, and the unroutable fraction shown in *Unrt* is with respect to this number.

Alexa. Blocking access to popular Internet sites or triggering alarms any time such sites are accessed would be disruptive to an enterprise. For my analysis, I periodically collected

the Alexa top 25 thousand domains (3–4 times a month) over the course of the measurement period [4]. To address the challenge that such lists can have significant churn [107], we restrict my whitelist to hold the *intersection* of all these top 25K lists (i.e., domains that were in the top 25K every time we polled Alexa over the 8-month measurement period), which left us with 12,009 domains. I then queried DNS for the A records, NS records and MX records of each domain, and collected the corresponding IP addresses. In total, I collected 42,436 IP addresses associated with these domains. I compute the intersection of these IPs with threat intelligence feeds and show the results in column *Alexa* in Table 3.3.

CDNs. CDN providers serve hundreds of thousands of sites. Although these CDN services can (and are) abused to conduct malicious activities [24], their IP addresses are not actionable. Because these are fundamentally shared services, blocking such IP addresses will also disrupt access to benign sites served by these IPs. I collected the IP ranges used by 5 popular CDN providers: AWS CloudFron [32], Cloudflare [31], Fastly [46], EdgeCast [42] and MaxCDN [83]. I then check how many IPs in threat intelligence feeds fall into these ranges. Column *CDNs* in Table 3.3 shows the result.

✦ **Finding:** Among the 47 feeds in the table, 33 feeds have at least one unroutable IP, and for 13 of them, over 1% of the addresses they contain are unrouteable. Notably, the Paid IP Reputation feed in the spam category has an unroutable rate over 78%. Although it is not documented, a likely explanation is that this feed may include unroutable IPs intentionally, as this is a known practice among certain spam feeds. For example, the Spamhaus DROP List [123] includes IP address ranges known to be owned or operated by malicious actors, whether currently advertised or not. Thus, for feeds that explicitly do include unroutable IPs, their presence in the feeds should not necessarily be interpreted as a problem with quality control.

I further checked feeds for the presence of any "reserved IPs" which, as documented in RFC 8190, are not globally routable (e.g., private address ranges, test networks, loopback and multicast). Indeed, 12 feeds reported at least one reserved IP, including four of the Paid IP Reputation feeds (excepting the spam category), six of the Badips feeds, and the FB Malicious

IPs and DShield IPs feeds. Worse, the Paid IP Reputation feeds together reported over 100 reserved IPs. Since such addresses should never appear on a public network, reporting such IPs indicates that a feed provider fails to incorporate some basic sanity checks on its data.

There are 21 feeds that include IPs from top Alexa domains, as shown in column *Alexa* in Table 3.3. Among these IPs there are 533 A records, 333 IPs of MX records and 63 IPs of NS records. The overlapped IPs include multiple instances from notable domains. For example, the IP addresses of www.github.com are included by Malc0de IP Blacklist. Paid IP Reputation in the malware category contains the IP address for www.dropbox.com. Alienvault IP Reputation contains the MX record of groupon.com, and Badips SSH also contains the IP addresses of popular websites such as www.bing.com.

Most of the feeds I evaluated do not contain IPs in CDN ranges, yet there are a few (including multiple Paid IP Reputation feeds, Badips feeds and Alienvault IP Reputation) that have significant intersection with CDN IPs. Alienvault IP Reputation and Badips feeds primarily intersect with Cloudflare CDN, while most of the overlap in the Paid IP Reputation malware category overlaps with AWS CloudFront.

Overall, the rate of false positives in a feed is not strongly correlated with its volume. Moreover, certain classes of false positives (e.g., the presence of Top Alex IPs or CDN IPs) seem to be byproducts of how distinct feeds are collected (e.g., Badips feeds tend to contain such IPs, irrespective of volume). Unsurprisingly, I also could find not correlation between a feed's latency and its accuracy.

3.2.7 Coverage

The coverage metric provides a quantitative measure of how well a feed captures the intended threat. A feed with perfect coverage would include all indicators that belong in a category. Unfortunately, as discussed above, there is no systematic way for evaluating the exact accuracy or coverage of a feed since it is unrealistic to obtain ground truth of all threat activities on the Internet.

However, there are some large-scale threat activities that are well-collected and well-studied. One example is Internet scanning. Researchers have long been using "Internet telescopes" to observe and measure network scanning activities [18, 41, 94]. With a large telescope and well-defined scan filtering logic, one can obtain a comprehensive view of global scanning activities on the Internet.

To this end, I collected three months of traffic (from January 1st to March 31st 2018) using the UCSD network telescope [120], which monitors a largely quiescent /8 network comprising over 16 million IP addresses. I then used the default parameters of the Bro IDS [19] to identify likely scanning traffic, namely flows in which the same source IP address is used to contact 25 unique destination IP addresses on the same destination port/protocol within 5 minutes. Given the large number of addressed being monitored, any indiscriminate scanner observed by threat intelligence feeds will likely also be seen in my data. Indeed, by intersecting against this telescope data we are able to partially quantify the coverage of each threat intelligence scanning feed.

The scanners I collected from the telescope consist of 20,674,149 IP addresses. The total number of IPs in all the scan feeds during this period is 425,286, which covers only 1.7% (363,799 shared IPs) of all the telescope scan IPs. On the other hand, telescope scanners intersect with 85% of all IPs in scan feeds. When looking at each feed, PA AlienVault IPs, DShield IPs Packetmail IPs, PA Lab Scan and PA Packetmail ramnode all have over 85% of their data intersected with telescope scanners; the other four, though, have less than 65% of their data shared (and the rate for PA Snort BlockList is only 8%).

To further understand how well each scan feed detects scanning activities, I measure how different sizes of scanners in the telescope are covered by each feed. Here, *scanner size* means how many IPs a scanner has scanned in the telescope within a day. Figure 3.3 shows the coverage rate of each feed over different sizes of scanners, ranging from 1,000 to 1 million. **Y** axis is the proportion of scanners of a given size or larger that are covered by each feed. (There are 7,212,218 scanners from the telescope whose sizes are over 1K, 271,888 that are over 100K and 17,579 are over 1 million.)

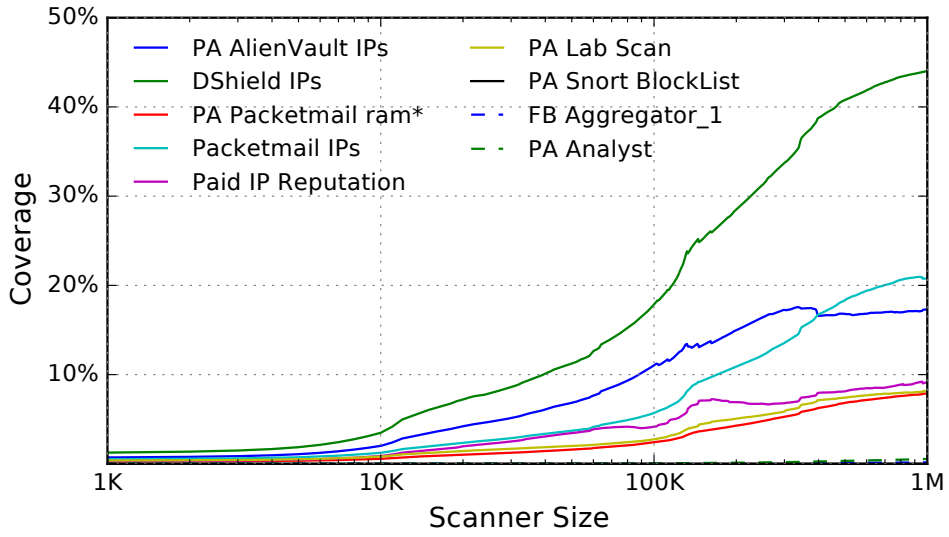


Figure 3.3. The coverage of each feed on different sizes of scanners.

✦ **Finding:** The union of all the scan IPs in the feeds covers less than 2% of the scanners collected by the telescope. Even if I only look at the scanners with sizes larger than 10,000, the overall coverage is still around 10%, suggesting the coverage capability of scan feeds is very limited. The graph shows that, as the scanner size increases, the coverage of each feed over the datasets also increases, and large feeds cover more percent of telescope scanners than small feeds. This trend aligns with the intuition that scan feeds tend to capture more extensive scanners.

It is surprising that the small scan feeds in my collection have a smaller percentage of their IPs shared with telescope scanners. This contradicts the idea that small feeds would contain a larger percentage of extensive scanners (that would most likely also be observed by the telescope).

3.3 File Hash Threat Intelligence

File hashes in a threat intelligence feed are indicators for malicious files. It is one of the most lightweight ways to mark files as suspicious. One can incorporate this data to block malicious downloads, malicious email attachments, and malware. Likewise, file hashes can be used to whitelist applications and these feeds can be used to ensure malicious files do not

appear in a customer's whitelist. In this section, I present the analysis on eight file hash feeds, also collected from December 1st, 2017 to July 20th, 2018. I use the same metrics defined in Section 3.1.3.

The file hash feeds I collected use a range of different hash functions to specify malicious files, including MD5, SHA1, SHA256 and SHA512 (and some feeds provided values for multiple different hash functions to support interoperability). Since most indicators in our dataset are MD5s, I have normalized to this representation by using other feeds and the VirusTotal service to identify hash aliases for known malicious files (i.e., which MD5 corresponds to a particualr SHA256 value).

I showed the detail result of the hash feeds analysis in Table 3.5 and Table 3.6. The second column group presents feed volume, average daily rate, the number of converted MD5s (Section 3.3.2) and exclusive proportion. *Not in VT* is fraction of hashes that are not found in VirusTotal, *Not det.* the fraction of hashes that are found in VirusTotal but are not labeled as malicious by any products, and *Detected* the fraction that are found in VirusTotal and are labeled malicious by at least one product. Column *Not in SD* shows the fraction of hashes in a feed that are not in Shadowserver Bin Check. *In NSRL* and *In AppInfo* show the absolute number of hashes found in Shadowserver (Section 3.3.3). *Exclusive* is based on the MD5-normalized hashes counted under *Converted*. All the other percentages in the table are based on *Volume*. I will explain about these result in detail in the followin sections.

Table 3.5. File hash feeds overview (Part I)

Feed	Volume	Avg. Rate	Converted	Exclusive
FB Malware	944,257	4,070	944,257	>99.99%
PA Malware Indicators	39,702	171	39,702	98.73%
PA Analyst	38,586	166	37,665	97.97%
PA Twitter Emotet	1,031	4.44	960	77.29%
PA OSINT	829	3.57	783	71.65%
PA Sandbox	298	1.28	115	95.65%
PA Abuse.ch	267	1.15	3	100%
PA Zeus Tracker	17	0.07	17	100%

Table 3.6. File hash feeds overview (Part II)

Feed	Not in VT	Not det.	Detected	Not in SD	In NSRL	In AppInfo
FB Malware	37.41%	50.50%	12.09%	99.89%	442	706
PA Malware Indicators	0.02%	0.04%	99.94%	>99.99%	2	0
PA Analyst	4.26%	2.82%	92.92%	99.95%	8	19
PA Twitter Emotet	11.74%	0.78%	87.49%	99.81%	0	2
PA OSINT	19.06%	0.84%	80.10%	99.88%	1	0
PA Sandbox	72.81%	0.34%	26.85%	100%	0	0
PA Abuse.ch	98.88%	0.75%	0.37%	100%	0	0
PA Zeus Tracker	88.24%	5.88%	5.88%	100%	0	0

3.3.1 Volume

File hashes, unlike IP threat data, are not transient—a file does not change from malicious to benign—and thus a far simpler volume analysis is appropriate. I report volume as the number of new hashes that are added to each feed during the measurement period.

As seen in Table 3.5, I examine each feed's volume and average daily rate. Like IP feeds, file hash feeds also vary dramatically in volume. The majority of the hashes are concentrated in three feeds: FB Malware, PA Malware Indicators, and PA Analyst, which also exhibit the highest daily rates. The other feeds are multiple order of magnitude smaller comparatively.

3.3.2 Intersection and Exclusive Contribution

As I mentioned earlier, to conduct intersection and exclusive analysis of file hash feeds, I need to convert indicators into the same hash type. Here I convert non-MD5 hashes into MD5s, using either metadata in the indicator itself (i.e., if it reports values for multiple hash functions) or by querying the source hash from VirusTotal [131] which reports the full suite of hashes for all files in its dataset. However, for a small fraction of hashes I am unable to find aliases to conver them to the MD5 representation and must exclude them from the analysis in this section. This filtering is reflected in Table 3.5, in which the Volume column represents the number of unique hashes found in each feed and the Converted column is the subset that I have been able to normalize to a MD5 representation.

✦ **Finding:** The intersections between hash feeds are minimal, even among the feeds that have multiple orders of magnitude differences in size. Across all feeds, only PA Analyst has relatively high intersections: PA Analyst shares 27% of PA OSINT's MD5s and 13% of PA Twitter Emotet's MD5s. PA Malware Indicators has a small intersection also with these two feeds. All other intersections are around or less than 1%. Consequently, the vast majority of MD5s are unique to one feed, as recorded in column *Exclusive* in Table 3.5. The "lowest" exclusivity belongs to PA Twitter Emotet and PA OSINT (still 77.29% and 71.65%, respectively). All other feeds showcase an over 95% exclusive percentage, demonstrating that most file hash feeds are distinct from each other.

Due to the different sources of malware between feeds, a low intersection is to be expected in some cases. For example, PA Twitter Emotet and PA Zeus Tracker should have no intersection, since they are tracking different malware strains. The other, more general feeds could expect some overlap, but mostly exhibit little to no intersection. Considering the sheer volume of the FB Malware feed, one might expect it would encapsulate many of the smaller feeds or at least parts of them. This is not the case, however, as FB Malware has a negligible intersection with all other feeds.

Due to the lack of intersection among the feeds, I omit the latency analysis of the hash feeds, as there is simply not enough intersecting data to conclude which feeds perform better with regards to latency.

3.3.3 Accuracy

Assessing the accuracy of file hash feeds presents a problem: there is no universal ground truth to determine if a file is malicious or benign. Thus, to gauge the accuracy of the feeds, I use two metrics: a check for malicious hashes against VirusTotal, and a check for benign hashes against Shadowserver's Bin Check service. Note that all the percentages discussed below are based on the *Volume* of each feed.

VirusTotal is a service that is often used when analyzing malware to get a base of

information about a suspected file. Anyone can upload a file to be scanned. Upon submission, these files will be scanned by more than 70 antivirus scanners, which creates a report on how many antivirus scanners mark it malicious, among other information. In this analysis, I query VirusTotal for the hashes in each file hash feed and then inspect the percent of hashes that are marked as malicious and how many AV scanners have recorded them. Due to the high volume of the FB Malware feed and the query rate limit of VirusTotal, I randomly sampled 80,000 hashes from the feed for this analysis.

Table 3.6 shows a breakdown of the base detection rates for each feed from VirusTotal. As the PA feeds decrease in volume, the rates at which they are found in VirusTotal also decreases. The larger PA feeds have a much higher detection rate than their smaller counterparts. On the other hand, FB Malware only has 37% of its data detected by antivirus scanners and 50% in VirusTotal with no detection despite being the largest feed. This could indicate that FB Malware focuses on threats that specifically target Facebook and that are not as relevant to most VirusTotal users, such as malicious browser extensions [37, 63, 65]. This might undermine the limited coverage of VirusTotal as an oracle to detect targeted threats that are not of broader interest.

To further understand how the scanners in VirusTotal report the feed's data, I plot a graph of what percentage of hashes in each feed are detected by how many VirusTotal scanners. As seen in Figure 3.4, each point means the proportion of indicators (Y value) in a feed that is detected by *over* X number of AV scanners in VirusTotal. Four feeds have more than 50% of their samples detected by over 20 scanners. PA Malware Indicators and PA Twitter Emotet did not experience a large detection drop before 35 scanners, indicating that most indicators in the two feeds are popular malicious files recognized by many AV vendors. While PA Sandbox has a large percent of its hashes not presented in VirusTotal, over 70% of its samples that are detected are marked by over 20 AV scanners, showcasing a high confidence detection.

To more fully gauge the accuracy of the file hash feeds, I also examined how each feed measured against Shadowserver's Bin Check Service [110]. The service checks file hashes against NIST's National Software Registry List (NSRL) in addition to Shadowserver's own

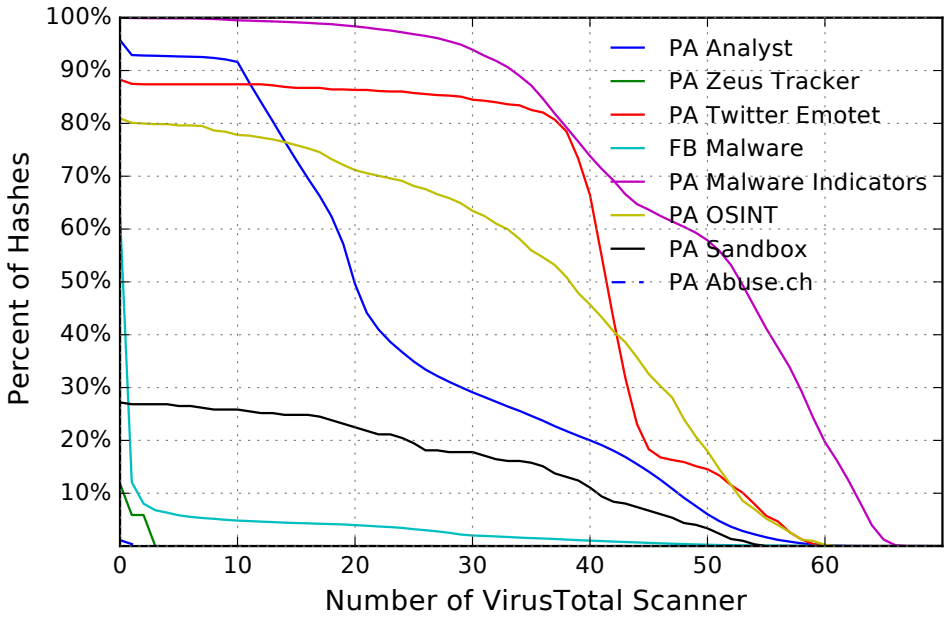


Figure 3.4. VirusTotal detection distribution.

repository of known software. Table 3.6 details how each feed compares with Shadowserver's Bin Check service.

It might be expected that there would be no hash found with Shadowserver's Bin Check service, but it is not the case. Some of the samples from the feeds that appear in Shadowserver are well known binaries such as versions of Microsoft Office products, Window's Service Packs, calc.exe, etc. In the event malware injects itself into a running process, it remains plausible that some of these well-known binaries find their way into threat intelligence feeds from users wrongly attributing maliciousness. While FB Malware has over one thousand hashes in Shadowserver, this is not a widespread issue, as all feeds have <1% of their hashes contained within Shadowserver's Bin Check service. This showcases that while there are a few exceptions, the feeds mostly do not contain well-known, benign files.

✦ **Finding:** Each PA feed has a negligible rate of occurrence within Shadowserver regardless of their VirusTotal detection, showing they do not contain generic false positives. Larger feeds exhibit high VirusTotal detection rates except for FB Malware, while small feeds have relatively low detection rates. This suggests that small hash feeds might focus more on

specific malicious files that are not widely known. FB Malware has a low VirusTotal occurrence despite its size and has over one thousand hashes in Shadowserver, but its overall low percentage of hashes within Shadowserver indicates that it does not contain many known files and might have threats not typically recognized by VirusTotal's scanners.

3.4 Longitudinal Comparison

In addition to the measurement period considered so far (December 1, 2017 to July 20, 2018), I also analyzed data from the same IP feeds from January 1, 2016 to August 31, 2016. These two measurement periods, 23 months apart, allow us to measure how these IP feeds have changed in two years. Table 3.7 and Table 3.8 summarizes the differences between these two measurement periods. *Avg. Rate* shows the percentage of daily rate changed over the old feeds. The two columns under *Unrt* show the unroutable rates of feeds in 2016 and 2018 separately. The two columns under *CDN* present the number of IPs fall in CDN IP ranges in old and new data. In the table, *2018* represents the current measurement period and *2016* the period January 1, 2016 to August 31, 2016.

Volume. As shown in Table 3.7 and 3.8, feed volume has definitely changed after two years. Among 43 IP feeds that overlap both time periods, 21 have a higher daily rate compared with 2 years ago, 15 feeds have a lower rate, and 7 feeds do not change substantially (the difference is below 20%). Volume can change dramatically over time, such as PA AlienVault IPs in the scan category which is 13 times larger than before. On the other hand, a feed like PA Blocklist Bots is now over 90% smaller.

Intersection and Exclusive Contribution. Despite the volume differences, the intersection statistics between feeds are largely the same across two years, with feeds in scan and brute-force having high pairwise intersections and feeds in other categories being mostly unique. Certain specific pairwise relations also did not change. For example, Badips SSH still shared over 90% of data in Dangerrulez Brute back in 2016, and Paid IP Reputation in malware was still the only feed that has a non-trivial intersection with multiple small feeds. Again, most data was

Table 3.7. Data changes in IP feeds compared against the ones in 2016 (Part I)

		Unroutable		*CDN*	
Feed	*Avg. Rate*	2016	2018	2016	2018
Scan Feeds					
PA AlienVault IPs	+1,347%	0.0%	0.0%	0	0
PA Packetmail ram*	+733%	<0.01%	<0.01%	0	0
Packetmail IPs	+135%	0.0%	0.0%	0	0
Paid IP Reputation	−57%	8.73%	1.65%	910	21
PA Lab Scan	−1%	0.0%	<0.01%	0	0
PA Snort BlockList	−97%	<0.01%	0.42%	1	0
FB Aggregator_1	+332%	0.0%	0.0%	6	0
PA Analyst	−44%	0.0%	0.41%	0	0
Botnet Feeds					
PA CI Army	+114%	<0.01%	<0.01%	0	0
Paid IP Reputation	−39%	0.63%	1.66%	15	74
PA Botscout IPs	+1%	0.01%	0.09%	1	0
PA VoIP Blacklist	+252%	0.0%	0.32%	0	0
PA Compromised IPs	−36%	0.10%	0.0%	0	0
PA Blocklist Bots	−95%	0.0%	0.0%	0	0
PA Project Honeypot	+63%	0.0%	0.0%	0	0
Brute-force Feeds					
Badips SSH	+30%	0.07%	0.19%	0	1
Badips Badbots	+1,732%	0.0%	1.04%	187	1,251
Paid IP Reputation	−62%	6.55%	0.03%	335	10
PA Brute-Force	−72%	0.0%	0.0%	0	0
Badips Username*	+3,040%	0.0%	0.53%	0	0
Haley SSH	+428%	0.04%	0.03%	0	0
FB Aggregator_2	+387%	0.12%	0.0%	0	0
Nothink SSH	+886%	0.56%	1.51%	0	0
Dangerrulez Brute	+0%	0.0%	0.0%	1	0
Malware Feeds					
Paid IP Reputation	−36%	0.18%	0.13%	15265	3,489
FB Malicious IPs	−77%	6.81%	2.14%	264	0
Feodo IP Blacklist	+0%	0.0%	0.0%	0	0
Malc0de IP Blacklist	−9%	0.0%	0.0%	132	11
PA Bambenek C2 IPs	+79%	0.0%	9.13%	0	0
PA SSL Malware IPs	−34%	0.0%	0.0%	0	0
PA Analyst	−93%	0.34%	0.0%	0	0
PA Abuse.ch*	−99%	0.49%	3.12%	0	0
PA Mal-Traffic-Anal	−53%	0.0%	0.51%	0	0
Zeus IP Blacklist	−66%	0.0%	0.0%	6	0

Table 3.8. Data changes in IP feeds compared against the ones in 2016 (Part II)

		Unroutable		*CDN*	
Feed	*Avg. Rate*	2016	2018	2016	2018
Exploit Feeds					
Badips HTTP	+326%	0.30%	0.67%	436	2,590
Badips FTP	+556%	0.01%	1.33%	0	2
Badips DNS	+9,525%	0.17%	0.50%	7	244
Badips RFI	+226%	0.0%	2.22%	0	0
Spam Feeds					
Paid IP Reputation	+133%	59.3%	78.7%	0	0
Badips Spam	+12,767%	0.0%	0.02%	0	0
Badips Postfix	−53%	<0.01%	1.29%	0	1
PA Botscout IPs	+18%	0.0%	0.06%	0	0
AlienVault IP Rep	+8%	0.57%	0.07%	479	1,040

exclusive to each feed two years ago: Across all six categories more than 90% of the indicators are not shared between feeds.

Latency. The latency relationship between feeds was also similar: timely feeds today were also timely two years ago, and the same with tardy feeds.

Accuracy. Feeds have more unroutable IPs now than before as shown in Table 3.7 and 3.8: In 2016, 22 of the 43 IP feeds had at least 1 unroutable IP; four feeds had unroutable rates over 1%. When checking the intersection with popular CDNs, the feeds that contain IPs in CDN ranges two years ago are also the ones that have these IPs today.

Shared indicators 2016–2018. I compared the data I collected from each feed in the two time periods, and found that 30 out of 43 feeds in 2018 intersect with their data from two years ago, and 9 feeds have an intersection rate over 10%. Three feeds in malware category, namely Feodo IP Blacklist, PA Abuse.ch Ransomware and Zeus IP Blacklist, have over 40% of their data shared with the past feed, meaning a large percent of C&C indicators two years ago are still identified by the feeds as threats today. Feeds in the botnet category, however, are very distinct from the past, with all feeds having no intersection with the past except Paid IP Reputation.

3.5 Absolute Latency

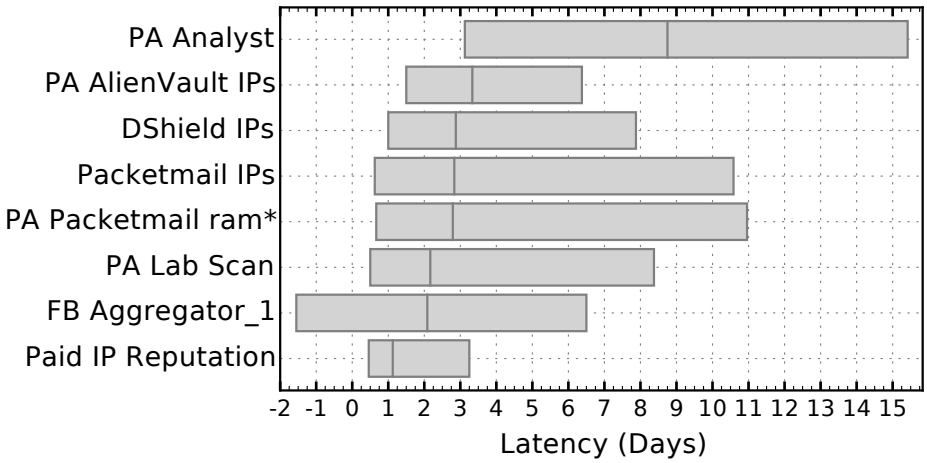
I defined the latency metric in this chapter as relative latency between threat intelligence sources, since it is easy to compute and allows consumers to compare feeds to each other on this aspect. However, it is also critical to know about the absolute latency distribution of indicators. Absolute latency represents how fast a feed can actually report a threat, which directly decides the effectiveness of the data when used in a pro-active way. As I already discussed in Section 3.2.5, absolute latency is hard to measure, as I do not have ground truth of the underlying threat.

In Section 3.2.7, I used an Internet telescope as the approximation for ground truth to measure the coverage of scan feeds. In Section 3.3.3, I used VirusTotal as an oracle to measure the accuracy of file hash feeds. Although these sources are not real ground truth and it is unclear how far away they are, these large and well-managed sources can help us, to a certain extent, profile the performance of threat intelligence feeds. In this section, I use these two sources again to approximate the absolute latency of indicators in scan IP feeds and malicious file hash feeds.

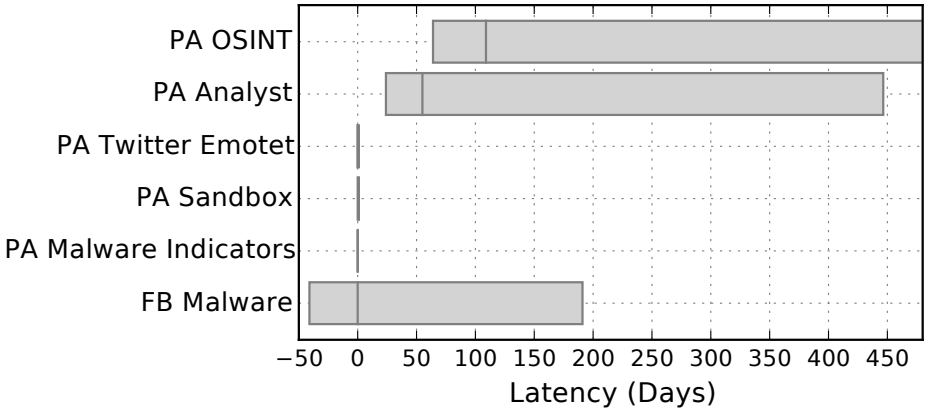
More specifically, I measure the latency of IPs in scan feeds relative to the first occurrence time of the same IP in the scanners collected from the telescope. Considering the massive size of the telescope, it should presumably detect scanners much sooner after the scanning event actually happened. I measure latency of file hashes relative to the `first_seen` timestamps queried from VirusTotal. The `first_seen` timestamp represents the time when the corresponding file is first uploaded to VirusTotal. VirusTotal is a very popular service and it is a convention for many security experts to upload new malware samples to VirusTotal once they discovered them. Therefore, this timestamp roughly entails when the security community first noticed the malicious file and can be a good approximation for absolute latency.

Figure 3.5 show the latency distribution of each feed, using the same plotting convention as in Section 3.2.5. Some feeds are not shown in the figure as there are too little data points in those feeds to reason about distribution.

✦ **Finding:** Comparing Figure 3.5a to Figure 3.2a, I can see that the median latency of



(a) Latency distribution in scan feeds relative to the Internet telescope



(b) Latency distribution in file hash feeds relative to VirusTotal

Figure 3.5. Distribution of indicators' latency in scan and file hash feeds. The scan feeds' distribution are calculated in hour granularity while the file hash feeds' distribution are calculated in day granularity.

feeds are all larger. This is consistent with my assumption that a large sensor tends to receive indiscriminate scanners sooner. Scan feeds' median lantecy are one to three days relative to the Internet telescope, except PA Analyst, whose median latency is almost nine days. The order of median latency between feeds changed compared with Figure 3.2a, but since the original relative median latencies among scan feeds are very close, the new order here is more likely to be statistics variances. Also, note that although the PA AlienVault IPs seems much slower than it is in Figure 3.2a, its 75 percentile latency is still the second smallest one.

On the other hand, the latency distributions of hash feeds vary more dramatically. PA Malware Indicators, PA Sandbox and PA Twitter Emotet are almost as fast as VirusTotal: all three feeds have 25 percentile and median latency equal to zero. PA OSINT and PA Analyst are comparatively much slower, and PA OSINT even has a 75 percentile latency of 1680 days. This might be because of the heterogeneous nature of malware feeds. The figure also shows that feed volumes do not imply their latency, as PA Analyst and FB Malware are much slower than the small hash feeds.

Figure 3.5 demonstrates that the Internet telescope and VirusTotal are indeed good approximations for absolute latency measurement, as most indicators in threat intelligence feeds are observed relatively later. However, every scan feed has over 2% of its indicators detected earlier than the telescope did. FB Aggregator$_1$ and DShield IPs even have over 10% of their indicators observed earlier. There is also a similar case in file hash feeds. This aligns with my observation in Section 3.2.5 that small feeds can still report a non-trivial amount of their data first. Another interesting observation is that both Facebook feeds, FB Aggregator$_1$ and FB Malware, have a large percent of their data observed earlier than the telescope or VirusTotal. This again suggests that Facebook (and its threat intelligence partners) might face more targeted threats, so those threats will be first observed by Facebook.

3.6 Discussion

3.6.1 Metrics Usage

Threat intelligence has many different potential uses. For example, analysts may consume threat data interactively during manual incident investigations, or may use it to automate the detection of suspicious activity and/or blacklisting. When not itself determinative, such information may also be used to *enrich* other data sources, informing investigations or aiding in automatic algorithmic interventions. I have introduced a set of basic threat intelligence metrics—volume, intersection, unique contribution, latency, coverage and accuracy—that can inform and quantify

each of those uses. Depending on a number of factors, such as the intended use case and the cost of false positives and negatives, some of these metrics will become more or less important when evaluating a threat intelligence source. For example, a feed with poor accuracy but high coverage might be ideal when an analyst is using a threat intelligence source interactively during manually incident investigations (since in this case, the analyst, as a domain expert, can provide additional filtering of false positives). Similarly, latency might not be a critical metric in a retrospective use case (e.g., post-discovery breach investigation). However, if an organization is looking for a threat intelligence source where the IPs are intended to be added to a firewall's blacklist then accuracy and latency should likely be weighted over coverage, assuming that blocking benign activity is more costly.

Another common real-world scenario is that a company has a limited budget to purchase threat intelligence sources and has a specific set of threats (i.e., botnet, brute-force) they are focused on mitigating. In such cases, the metrics I have described can be used directly in evaluating threat intelligence options, biasing twoards sources that maximize coverage of the most relevant threats while limiting intersection.

3.6.2 Data Labeling

Threat intelligence IP data carries different meanings. To properly use this data, it is critical to know what the indicators actually mean: whether they are Internet scanners, members of a botnet or malicious actors who had attacked other places before. I have attempted to group feeds by their intended meaning in my analysis.

However, this category information, which primarily comes from threat intelligence sources themselves, is not always available. Feeds such as Alienvault IP Reputation and Facebook Threat Exchange sources contain a significant number of indicators labeled "Malicious" or "Suspicious." The meanings of these indicators are unclear, making it difficult for consumers to decide how to use the data and the possible consequences.

For feeds that provide category information, it is sometimes too broad to be meaningful.

For example, multiple feeds in my collection simply label their indicators as "Scanner." Network scanning can represent port scanning (by sending SYN packets), or a vulnerability scan (by probing host for known vulnerabilities). The ambiguity here, as a result of ad-hoc data labeling, again poses challenges for security experts when using threat intelligence data.

Recently, standard threat intelligence formats have been proposed and developed, notably IODEF [59], CybOX [36] and STIX [117], that try to standardize the threat intelligence presentation and sharing. But these standards focus largely on the data format. There is room to improve these standards by designing a standard *semantics* for threat intelligence data.

3.6.3 Limitations

There are several questions that this study does not address. I attempted to collect data from a diverse set of sources, including public feeds, commercial feeds and industrial exchange feeds, but it is inherently not comprehensive. There are some prohibitively expensive or publication-restricted data sources that are not available to us. More specialized measurement work should be done in the future to further analyze the performance of these expensive and exclusive data sources.

A second limitation is my visibility into how different companies use threat intelligence operationally. For a company, perhaps the most useful kind of metric measures how a threat intelligence source affects its main performance indicators as well as its exposure to risk. Such metrics would require a deep integration into security workflows at enterprises to measure the operation effect of decisions made using threat intelligence. This would allow CIOs and CSOs to better understand exactly what a particular threat intelligence product contributes to a company. As a researcher, I do not use threat intelligence operationally. A better understanding of operational needs would help refine my metrics to maximize their utility for operations-driven consumers.

The third limitation is the lack of ground truth, a limitation shared by all the similar measurement work. It is simply very difficult to obtain the full picture of a certain category

of threat, making it very challenging to precisely determine accuracy and coverage of feeds. In this study, I used data from an Internet telescope and VirusTotal as a close approximation. There are also a handful of cases where a security incident has been comprehensively studied by researchers, such as the Mirai study [11], and such efforts can be used to evaluate certain types of threat intelligence data. But such studies are few in number. One alternative is to try to establish the ground truth for a specific network. For example, a company can record all the network traffic going in and out of its own network, and identify security incidents either through its IDS system or manual forensic analysis. Then it can evaluate the accuracy and coverage of a threat intelligence feed under the context of its own network. This can provide a customized view of threat intelligence feeds.

3.7 Acknowledgement

This chapter, in part, is a reprint of the material as it appears in Proceedings of the Usenix Security 2019. *Reading the Tea Leaves: A Comparative Analysis of Threat Intelligence.* Vector Guo Li, Matthew Dunn, Paul Pearce, Damon McCoy, Geoffrey M. Voelker, Stefan Savage, Kirill Levchenko. The dissertation author was the primary investigator and author of this paper. I really like to thank my coauthors on this paper, without who I will never be able to finish this work. I like to especially thank Paul Pearce, who helped me a lot throughout the project. At the beginning, when I had little idea about what to do and was so unfamiliar with data analysis tools, he gave me critical hands-on assists. I am also very grateful to Alberto Dainotti and Alistair King for sharing the UCSD telescope data and helping me with the analysis, also professor Micheal Bailey for sharing the Mirai Botnet data. Besides, I like to thanks professor Aaron Schulman, who encouraged me a lot when the first submission of this paper was rejected, and helped me regain my confidence in the work.

Chapter 4

Threat Intelligence Uses

In the previous chapter, I talked about the data characteristics of existing threat intelligence products and their limitations. Given the current status of threat intelligence, another side of the coin is how people are actually using the data currently. This question belongs to the empirical study of data usage, as I listed in the introduction chapter.

As one can see, threat intelligence data can be used in a number of different ways. It can be used forensically during incident response (i.e., to better understand and attribute a threat after it has gained access to a network), it can be used reactively to generate alerts of suspicious activity in a SIEM system (i.e., to raise awareness of a potential threat that is currently active), or it can be used proactively to block traffic (and hence block the associated threats). This last category of action, traditionally called "blacklisting", is uniquely attractive to a defender since, if effective, it can foreclose certain threats without requiring individualized attention from a human analyst. Indeed, it is common to see such precise scenarios highlighted in the marketing materials for virtually all threat intelligence offerings.

However, despite all the promises, it is far from clear how people actually adopt threat intelligence data, especially for proactive traffic blocking. Proactively blocking traffic based on threat intelligence data is a strong action, and as I showed in last chapter, threat intelligence feeds can be far from comprehensive and may include significant numbers of false positives. This could cause an organization to inadvertently block benign Internet sites. Moreover, on a broader

scale, a mistakenly added IP in a threat intelligence feed can effectively be denied service from all organizations using that feed to block traffic. Given this, it is important to understand the extent to which network administrators are willing to use such third-party data to block network traffic.

In this chapter, I will take the first step towards understanding this question. In particular, this study seeks to better understand the extent to which network administrators make use of threat intelligence IP feeds (more commonly known as IP blacklists) to proactively block network traffic and, if they do, which kinds of data sources they are using for that purpose.

The principal challenge in pursuing this question is that such decisions are largely invisible: a network choosing to block IP address x, is externally indistinguishable from one that does not, except to the owner of IP address x. Moreover, for operational security reasons, few organizations are willing to publicly document the details of their network defenses. Thus, there is no simple mechanism to determine if a network blocks certain traffic, let alone a means to determine the data source driving such a decision.

In this chapter, I explore this question via a combination of inference and careful testing, resulting in three primary contributions. First, building on prior work designed to detect censorship [43, 97], I develop, test and validate an inference technique using the IP ID side-channel to detect network-layer blacklisting. Second, by using this technique with a carefully chosen set of IP addresses, I am able to attribute blocking actions to the use of particular blacklists. Finally, I conduct a large-scale pilot study covering over 220K U.S. hosts to explore the diversity in blacklisting behaviors. Together, this work uncovers the use of 9 popular public IP blacklists among the hosts I surveyed, and demonstrate relations between these hosts and their update pattern on blacklists. I further investigate a broader use of blacklists among the hosts, and discovered over 73K hosts has shown blacklist related blocking behavior.

4.1 Related Work

4.1.1 Internet Connection Blocking

In this study, I focus on one specific way of how organizations use threat intelligence data: use threat intelligence IP feeds as direct rule-set to block network traffic. This behavior is very similar to other forms of Internet connection blocking, notably Internet censorship, geo-blocking and Tor blocking.

Previous works have studied Internet censorship [12, 95, 7, 141, 30], geo-blocking [58, 85, 3], and Tor blocking [112, 66]. However, these measurement studies all rely on having vantage points in the target regions, so the researchers can directly measure the network effects and acquire the results. There are several public projects that facility such studies by providing access points around the global, like RIPE Atlas [104], OONI [90] and ICLab [57]. For Tor blocking, it is also not hard for the researchers to get access to a Tor exit node [66], then conduct the following measurement from that node. But these studies are restrained by the vantage points they could get, as it is hard to get these vantage points in large quantities, and these points are also heavily biased towards certain networks. For country-wide censorship or geo-blocking measurement, this is not a big issue. But in this case, since I want to conduct large scale measurements over a broad set of online hosts, it is very difficult to acquire vantage points that can meet my requirements.

Recent work by Ensafi *et al.* [43] and Pearce *et al.* [97] demonstrated the method to use IP ID side channel to measure Internet connectivity. This is an indirect method that allows an observer to measure the connectivity between two hosts without having access to either of the hosts. In this study, since I do not have access to either the hosts on blacklists, this method is an ideal method for this measurement. But I need to modify the original method to better suit this experiment, and I also need to take extra effort to eliminate the interfering signals. I will establish on the methodology in more detail in Section 4.2.

4.1.2 IP ID Side Channel

The Identification (ID) field of an IPv4 packet is a 16-bit value in the IP packet header. It is designed primarily to support IP packet fragmentation and reassembly. If a large IP packet is fragmented when sent through the network, the receiving end will use this field to identify the fragments from the same IP packet and re-assemble them. This requires the 16-bit value to be unique for every datagram of a given source address, destination address, and protocol, such that it does not repeat within the maximum datagram lifetime [99].

One easy way to implement the IP ID field for networking is to use a global counter. In this case, a system uses one 16-bit variable to set the IP ID value for all the IP packets it sends out, and increments the variable by 1 after every packet. This simple solution ensures the IP ID value of all packets are unique, and it is how many early systems implemented their IP ID mechanisms [70].

This implementation create a side channel that allows anyone without access to a host to observe the traffic volume from that host. An observer, by probing the host twice separated by some time interval and checking the corresponding IP ID increase, can learn about the number of packets the host sent out during this period. This side channel can be used to observe many different network effects, like host alias detection [116], where observers can detect if two online hosts actually correspond to one physical machine by monitoring their IP ID changes, and NATed host counting [17], where observers can identify the number of machines behind a NAT by tracking the number of IP ID sequences from the packets comming out of the network.

In order to eliminate this side channel, new systems implement the IP ID generation by using different counters for different traffic, so different observers will see a different IP ID sequences from the same host. However, there are still a significant number of hosts on the Internet that still use the global counter implementation. For example, Windows 8 and older still use this implementation [70]. These hosts give us sizable amount of candidates to conduct this measurement.

4.2 Methodology

In this chapter, I use the IP ID side channel (Section 4.1.2) to determine whether a particular online host blocks traffic using one or more known blacklists. In brief, I begin by identifying hosts that are suitable for this technique. I refer to such hosts as *reflectors*. Then I randomly sample a set of IP addresses from IP blacklists that could be used to block traffic. Such IP addresses will be refered as *blacklist IPs*. Then I measure if each reflector blocks packets whose source addresses are blacklist IPs. If one reflector blocks all IP addresses I sampled from one particular blacklist, then it is probably using that blacklist.

In this section, I first describe how the technique works at a high-level (Section 4.2.1). Then I detail the method for identifying proper reflectors for the measurement (Section 4.2.3), how to choose target blacklists to test (Section 4.2.4), and how to sample IP addresses from each blacklist (Section 4.2.5). In Section 4.2.6, I explain the experiment design in detail and how it works in real world scenarios. Section 4.2.7 further describes the steps I take for sanity check, and in Section 4.2.8 I discuss the ethical considerations of the methodology.

4.2.1 Technique Overview

To measure if one reflector is blocking one particular IP from a blacklist, I send a train of packets (here I use SYN-ACK packets) from the measurement machine to the reflector. The packet train consists of packets whose source address is the blacklist IP (spoofed), bracketed by packets whose source address is the measurement machine, as illustrated in Figure 4.1. If a firewall in the reflector's network blocks packets from the blacklist IP, the reflector will not receive packets with the blacklisted source address. In this case, it will only receive packets with the measurement machine's source address. On the other hand, if there is no firewall blocking such packets, the reflector will receive the entire packet train.

In an ideal world, where there is no packet loss during transmission and no extra traffic on the reflector, one would see a clear difference in the IP IDs from the responses between the two different scenarios. In particular, I expect the reflector to send a RST response for each SYN-ACK

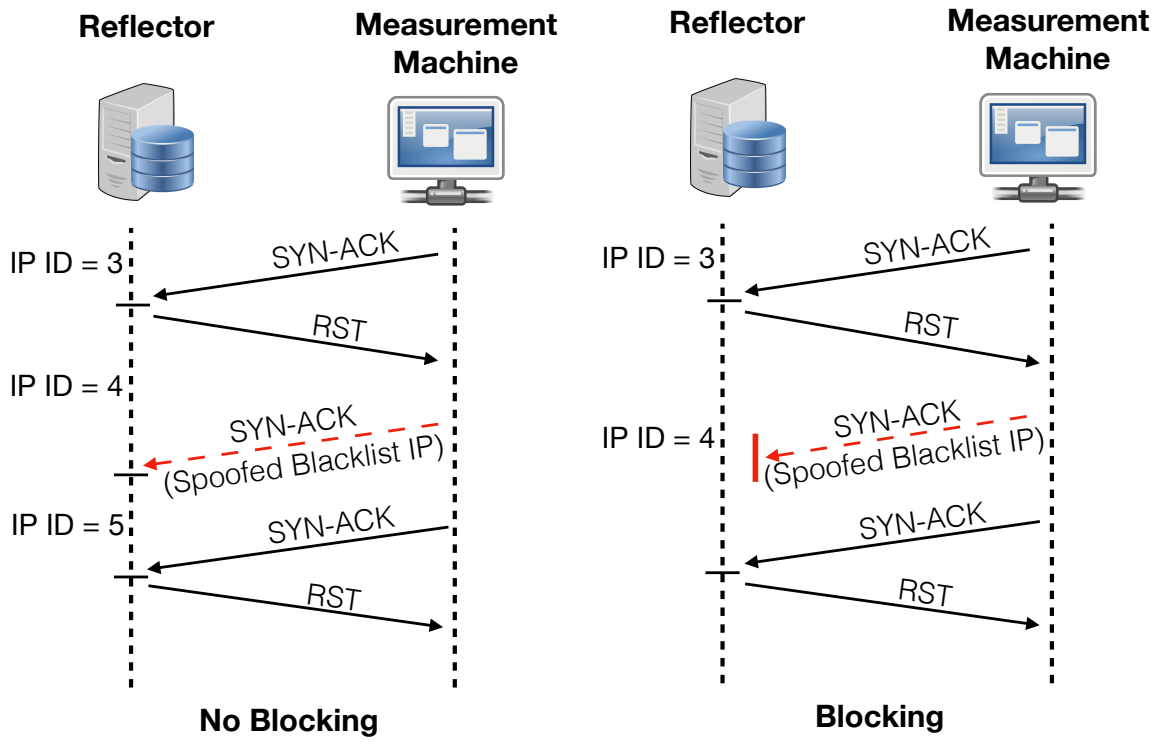


Figure 4.1. The basic method to detect network blocking using the IP ID side channel.

packet I send, and I will receive the responses for the SYN-ACK with the measurement machine's source address. The IP IDs of these received RST packets will reflect the number of packets sent by the reflector. If the reflector did not receive the SYN-ACK packets with the blacklist IP as source addresses(because they were blocked by a firewall), the IP ID sequence in the RST responses will be an increasing sequence without gaps (the "Blocking" case in Figure 4.1). On the other hand, if the reflector did receive the SYN-ACK packets with the blacklist IP, it would have sent a RST packet in response to each such packet, incrementing the IP ID counter each time. While I will not see the RST packets sent to the blacklist IP, I *will* observe the increments in the IP ID sequence. More specifically, one would see a gap in the IP ID sequence of packets received by the measurement machine (the "No Blocking" case in Figure 4.1). These two cases, illustrated in Figure 4.1, allow us to determine whether a particular blacklist IP is blocked by some network device, such as a firewall, somewhere between the measurement host and the

reflector. When there is no blocking in place (left), the measurement machine will see an IP ID gap in two RST responses: the second IP ID will increase by two. Whereas if there is network blocking (right), then the two IP IDs will not have a gap: the second IP ID will only increase by one.

This measurement technique is inspired by the method proposed in previous work [43, 97]. However, to make it work for my objective I need to handle a few more important issues, as I will see in the rest of this section.

4.2.2 Comparison with Previous Method

At a high level, the objective of the measurement is to determine, from a third point, if a reflector blocks a blacklist IP. The blocking behavior that we want to measure is *inbound blocking* – that is the incoming traffic is blocked and as a result traffic does not reach the intended host. A typical example is a network firewall, where it can stop certain network packets from reaching hosts behind the firewall. Here I abstract the reflector as $Host_A$, blacklist IP as $Host_B$, and the problem is to detect the connectivity between two hosts on the Internet.

In previous works [97, 43], to measure the connectivity between two hosts from a third party, they send spoofed packets to impersonate one of the hosts. This methodology – one I refer to as the *triangle measurement* takes advantage of the TCP 3-way handshake protocol, as shown in Figure 4.2.

The measurement machine first sends a probe to the $Host_A$, in this case a TCP SYN-ACK packet. $Host_A$ responds with a RST packet since it received a SYN-ACK without the preceding SYN packet. Thus, the measurement machine gets the first IP ID $IP\text{-}ID_1$ from the RST packet (corresponds to Step 1 in the figure). Next, the measurement machine sends a spoofed TCP SYN packet to $Host_B$, with source IP address set to the IP address of $Host_A$ (Step 2). $Host_B$ then sends a responding SYN-ACK packet to $Host_A$, which causes $Host_A$ to respond with a RST packet (Step 3), and increment its IP ID counter by 1. Finally, the measurement machine probe $Host_A$ again and get the second IP ID $IP\text{-}ID_2$ (Step 4).

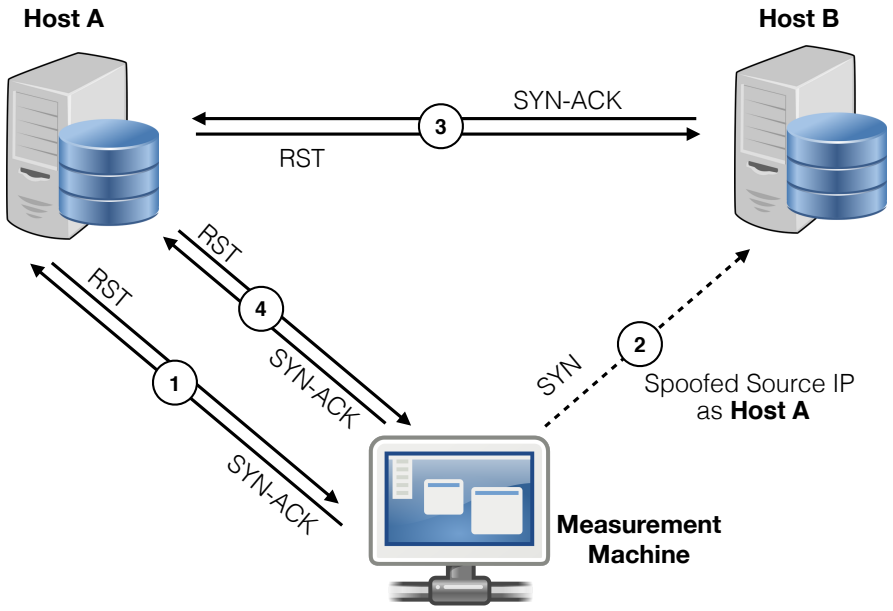


Figure 4.2. Measurement method used in previous work.

Now I can infer whether $Host_A$ is inbound blocking traffic from $Host_B$ by observing the difference between $IP\text{-}ID_1$ and $IP\text{-}ID_2$. Assuming there is no packet loss, and that $Host_A$ does not have any extra traffic besides my measurement traffic, then $IP\text{-}ID_2 = IP\text{-}ID_1 + 2$ implies there is no inbound blocking, since it indicates that $Host_A$ received both packets in step 3 and step 4, while $IP\text{-}ID_2 = IP\text{-}ID_1 + 1$ means there is blocking.

Previous work chose this "triangle measurement" because it ensures that in step 3, the packets from $Host_B$ will go through the same routes as the traffic originated from $Host_B$. So from $Host_A$'s perspective, it can not identify that the traffic from $Host_B$ are spoofed. However, in this schema, one hard requirement is that $Host_B$ needs to be active and responding to TCP SYN probe. This is not an issue for censorship measurement, as $Host_B$ in this case are popular sites (Google, Facebook, Twitter etc.) that guaranteed will respond to SYN probe.

However, in this case, I need to measure whether $Host_A$ is blocking traffic from a blacklist IP($Host_B$). But there is no guarantee that these IP addresses are active and thus may not respond to the SYN probe. In fact, we found that the percentage of responding IPs in a blacklist can be as

low as less than 20%. This dramatically reduces the candidate IPs I can sample from a blacklist to test, especially for small blacklists that only have a few hundred IPs. Furthermore, there are many other additional constraints when shortlisting IPs for measurement from a blacklist. The requirement that blacklist IPs respond to SYN probe does not work for this use case.

In order to get around this limitation I adjust the measurement methodology. In this new methodology, the measurement machine directly sends spoofed packets to the target host, as shown in Figure 4.3. In this case, the measurement machine first probes $Host_A$ to get the first IP ID $IP\text{-}ID_1$, then it sends a spoofed packet, with source IP set to $Host_B$(Blacklist IP), directly to $Host_A$. Finally, it sends a second probe to $Host_A$ and get the second IP ID $IP\text{-}ID_2$. Now I can use the same logic as before to infer whether $Host_A$ is inbound blocking $Host_B$. In this approach, I do not require $Host_B$ to be actively responding SYN packets, any IP address can be used here to conduct the test. The drawback is that the spoofed packets now at times go through a different route versus the packets originated from $Host_B$. Some network that implement spoofed packet detection [47] could drop the spoofed packets, giving us a false signal of inbound blocking. Therefore, when selecting hosts, I conduct extensive tests to weed out hosts that have such detection logic in place. We find that not a lot of target hosts have such detection logic. I will talk in detail about host selection in the follow sections.

The disadvantage of this approach is that we cannot detect outbound blocking — wherein the spoofed packet reaches the reflectors but the responses are blocked when going out of the network [97]. Based on our experience talking with several security companies, most customers deploy inbound blocking or bi-directional traffic blocking, so we do not think missing outbound blocking is a major concern.

4.2.3 Finding Suitable Reflectors

At a high level, my methodology relies on being able to infer blocking using the IP ID side channel. Keeping that in mind, listed below is the criteria I look for when scanning the Internet to search for suitable reflectors.

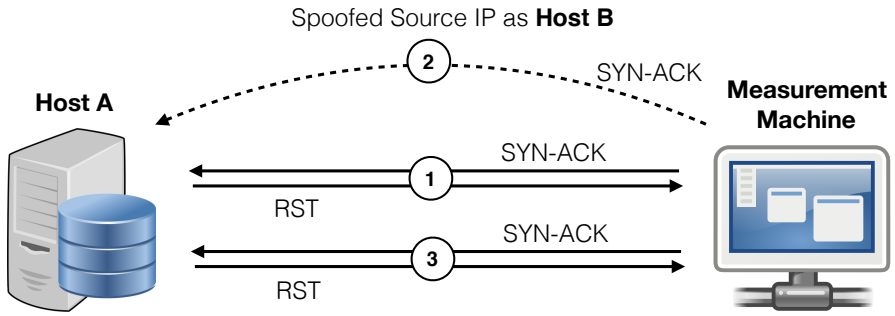


Figure 4.3. Measurement method used in this work.

- **RST packet generation:** The reflectors must reply with a RST packet to a TCP SYN-ACK packet without an established connection. Some hosts drop incoming SYN-ACK packets if there is no corresponding SYN packet. These hosts are not suitable for my measurement methodology. I choose SYN-ACK packets instead of SYN because it does not create an intermediate state on the reflectors and connection is terminated in one go.

- **Shared monotonic increasing IP ID counter:** The IP ID counter in the reflector needs to be globally shared, so all network traffic generated from the host will use the same counter for IP ID assigning. It also needs to be monotonically increasing, so I can observe the number of packets generated by the host between two measurements using the difference of IP IDs.

- **Low traffic:** The technique requires the host to have low traffic volumes in general, since the technique depends on the fact that I can observe a clear difference in IP ID increases when sending spoofed packets. If there are always many other packets coming to the host, it would be infeasible to observe the IP ID changes triggered by the measurement packets.

- **No ingress filtering:** I send spoofed packets to reflectors to infer traffic blocking. However, some network providers utilize ingress filtering techniques and drop packets once they detect the packets are not from the networks they claimed to originate. This would cause the spoofed packets being dropped and give us a false signal of traffic blocking.

- **No stateful firewall blocking:** Some networks deploy a stateful firewall that blocks access from a source IP after receiving too many repetitive packets. One example is to defend against SYN floods [75]. While I try to keep the number of the measurement packets as low as possible, if the spoofed packets trigger such firewall rules and then I are blocked by the firewall, I will incorrectly conclude that the reflector uses a blacklist to block that IP.

I try to uncover how online hosts are using IP blacklists to block traffic. But when looking at the problem on a global scale, there are many policy related reasons why a host blocks network traffic, such as censorship. These alternate sources of blocking could disrupt the measurement, making it hard to distinguish the type of security-related blocking I target. To simplify the problem, in this chapter I focus on the hosts in United States.

I find reflectors in the United States with open ports using a snapshot of Censys [25] scanning data from November 8, 2019. Then I scan these 40 million hosts to identify the ones with the IP ID side channel. I send multiple probes to each host targeting an open port from different source addresses, and then check IP IDs in the responses. In the case where hosts have multiple open ports, I randomly select a port to send the probe.

To identify stateful firewalls, I send SYN-ACK packets to each reflector in two different patterns: 1 second per packet with 24 packets, and 5 packets per second with 60 packets, which corresponds to the speed I probe reflectors during actually experiments (see the following section). I repeat each experiment 6 times and discard the hosts that block us after the experiment. To find the hosts with low extra traffic, I send 24 probes to each host, 1 per second, and repeat the experiment 5 times at different times of the day. I then collect the result and only select the hosts where more than 30% of their IP ID increases are equal to 1 per second — that is, the host did not receive any extra traffic besides my probes, and all of the increases were smaller than 10 within a second.

Finally, I try to identify hosts experiencing ingress filtering. To account for differences in ingress filtering that may possibly occur on different network paths to the reflectors, I acquired 7

Table 4.1. The number of reflectors (IP addresses) identified in the United States, and the corresponding count of /24 prefixes and Autonomous Systems.

Category	Count
IP Addresses	222,782
/24 Count	128,712
Autonomous Systems	3,371

vantage points around the world to exercise different paths. These vantage points are from the US west coast, east coast and midwest, and places in Asia, Europe, Australia and South America.

I then send spoofed packets from my measurement machine to the reflectors with spoofed source addresses corresponding to the 7 vantage points, and later collect responses at each vantage point. I only select the reflectors that send responses to all 7 vantage points, meaning they did not drop spoofed packets on these network paths.

Eventually, I identified 222,782 IP addresses in US that meet the requirements.[1] For the purpose of this chapter, here I treat each individual IP address as a distinct reflector. Detailed numbers are presented in Table 4.1.

By construction, the set of reflectors we use will necessarily have certain biases. To understand what fraction of networks of potential interest to others this might cover, we queried the Alexa top 100K domains as of Dec. 17th, 2019 for their A records and MX records and obtained their corresponding IP addresses. Of these, we identified a total of 94,846 IPs that are located in the US, covering 34,083 /24s. While no attempt was made to find reflectors of these networks *a priori*, our selection methodology identified at least one reflector in 16.9% of these /24s. When only looking at the top 10K domains, our data set covers 13.2% of US /24s.

We also checked the WHOIS record of each reflector and identified all hosts associated with education institutions. In total, our data set includes 4,370 education IPs, ranging across 181 different institutions, and covers 40 out of the top 100 US universities based on the US News ranking [129]. Thus, while there may be networks without a suitable reflector for one reason or

[1] I initially discovered more than 300K reflectors, but during my experiments some hosts became inactive. The number reported here is the final number after I finished all the experiments.

Table 4.2. Top 9 popular public IP blacklists.

Blacklist	Average Number of IPs
Spamhaus DROP	$\sim$ 20,000,000
Spamhaus Don't Route Or Peer Lists	
Spamhaus EDROP	$\sim$ 900,000
An extension of the Spamhaus DROP list	
DShield Top Blacklist	5,120
DShield.org recommended top 20 /24s to block	
ET Compromised	$\sim$ 400
EmergingThreats.net recorded compromised hosts	
Snort IP Filter List	$\sim$ 1,500
labs.snort.org supplied IP blacklist	
BDS IP Ban List	$\sim$ 6,000
Binary Defense System ban list	
Feodo IP Blacklist	$\sim$ 700
Abuse.ch Feodo tracking list	
Blocklist De Blacklist	$\sim$ 30,000
Blocklist.de blacklist IPs	
Tor IP Blacklist	$\sim$ 6,000
IPs that belong to Tor network (not just exits node)	

another, our technique is applicable to a large number of existing networks.

4.2.4 Choosing the Blacklists

I choose candidate blacklists from public IP blacklists since I do not have access to commercial blacklists. In this work, I use the FireHOL IP blacklist collection [49] which aggregates over 100 public IP blacklists. However, I cannot reasonably test against all the blacklists and so, for the purposes of this chapter, I would like to select the most popular public IP blacklists and then do a more detailed measurement of them.

For each of the public IP blacklists, I sample five IPs (using the criteria in Section 4.2.5) from each list and test how many reflectors block all sampled blacklist IPs in each blacklist (using the method in Section 4.2.6). With this experiment, I generate a list of the most popular blacklists. Of course, five sample points from one list is not a strong enough indicator to conclude whether a host is using that blacklist or not. That said, the goal of this measurement is not to identify the

exact hosts that use each blacklist. Rather, it is estimate how widely used these blacklists might be so that I can use them for more detailed measurements. I repeat the measurement twice and select the top 9 popular blacklists,[2] as listed in Table 4.2.

Note, the Tor IP Blacklist is the combination of three Tor blacklists, as they mostly have the same content. This is primarily done since the huge overlap between the three lists means that I have very few blacklist IPs that meet my exclusive criteria (see Section 4.2.5). The Tor IP Blacklist essentially includes IPs for all nodes in the Tor network, including entry nodes, so the reflectors who block IPs on this list can neither be accessed from Tor nor use Tor services themselves.

4.2.5 Sampling Blacklist IPs

For determining if a reflector uses a particular IP blacklist, I use a sample of IPs from the blacklist to test since it is infeasible for us to test all blacklist IPs. Further, to obtain a definitive signal from my measurement, I adhere to the following constraints when sampling blacklist IPs:

- **Exclusive:** A blacklist can share part of its contents with other blacklists. To reasonably infer whether a reflector is using a specific blacklist, I need to test with the IPs that are unique to that blacklist — IPs that are only in this blacklist, but no others.

- **Stable:** The IPs on a blacklist change over time. To reliably measure if a reflector blocks IPs from a certain blacklist, I need the sampled IPs to stay in the blacklist throughout the measurement. I discard any measurements where the blacklist IP does not remain on the blacklist for the duration of the measurement.

- **Routable:** IP blacklists can contain unroutable IPs [77]. Sending packets with an unroutable source address results in a large portion of packets being dropped (which could potentially happen at the end ISP or the transient link). Packet drops due to unroutable IPs

[2] I initially selected 10 blacklists. However, one blacklist, abuse.ch Ransomware List, was discontinued by the provider during my experiments, and so I removed that blacklist from consideration.

would create noise in the measurement. Therefore, when sampling IPs from blacklists I ensure that the IPs are routable.

- **Geo-location diversified:** Besides blacklisting, another common reason for a host to block certain traffic is geo-blocking, where a host blocks all traffic coming from a certain country or a certain region. To minimize the effect of geo-blocking, I prioritize IPs that are from the United States when sampling IPs. The assumption is that a host in the US will not block traffic from its own country. For IPs from other countries, I try to increase the diversity of IP locations, making sure these IP are not concentrated in a few countries when possible.

- **Not from the hosts' network (AS disjoint):** I observed many networks drop spoofed packets when the spoofed source addresses are within their own network. So when selecting IPs, I make sure that these IPs are not from the same ASes as one of the reflectors.

To obtain "exclusive" IPs from a blacklist, I would potentially need an "oracle" that includes all IP blacklists, which is impractical. In this work, I use the public IP blacklists collected by FireHOL, as mentioned earlier, to calculate the exclusive part of each target blacklist. For "stable" IPs, I collect all the target blacklists hourly, and ensure the sampled IPs are in the blacklist through the duration of the experiment. To satisfy the "routable" requirement, I use the daily RouteView data [128] to identify BGP routable IPs. As for geo-location diversity, I use Netacuity [88] to make sure for each experiment the sampled IPs cover as many different countries as the data allows.

4.2.6 Experiment Design

Previously, I described the ideal model of the measurement method, which explained the idea and workflow at a high-level. However, this model does not take into consideration packet loss or extra traffic at the reflector. As one would expect, these assumptions are unrealistic in a real world scenario, as packet loss can happen at many stages along the path. Furthermore, there is no guarantee that the host with an open port online will not receive any other traffic. Thus, to

perform the measurement in the real world, I need to take all these factors into consideration and make sure that the analysis model is robust to these real world uncertainties.

Moreover, the detection methodology also needs to be *efficient*, *accurate*, and have *low overhead*. Since I need to measure every pair of `(reflector, IP)`, which is a very large number, and the blacklist content changes rapidly, the detection method needs to be efficient so that I can finish the measurement in a reasonable amount of time. The method should also have a low false positive and false negative rate, so I can be confident about the result. Finally, it should require as few packets as possible, both to meet network bandwidth limitations on the measurement machine side and reduce impact on reflectors.

I define a *trial* as a single measurement that tests if a reflector blocks a blacklist IP. Figure 4.4 shows the process of one trial in detail. The solid blue lines are the *probe packets*. Dashed red lines are the *spoofed packets*. The spoofed packets impersonating blacklist IPs trigger the increase of the reflectors' IP ID. The IP ID of responses to probe packets are used to determine blocking behavior. For each trial, the measurement machine sends five consecutive *probe packets* to the reflector, with each packet being sent one second apart. In the experiment, the probe packets are TCP SYN-ACK packets and I get IP IDs from response RST packets. Between the third and fourth probe packets, the measurement machine sends five *spoofed packets*, also TCP SYN-ACK, with source IPs equal to the blacklist IP. And between the fourth and the fifth probe packets, it sends another five spoofed packets. Each time I send the five spoofed packets, I send them 0.15 second apart consecutively, spreading them across the one-second window between two IP ID probes.

Now, I inspect the increases between the IP IDs for the packets received by the measurement machine. In an ideal world, when there is no other traffic generated by the reflector, and no packet loss during the measurement, one should observe that the IP ID increases between consecutive probes by exactly 1, and for the last two deltas, since I send the spoofed packets in between the probe packets, the final IP ID increases will be different based on the host's blocking behavior.

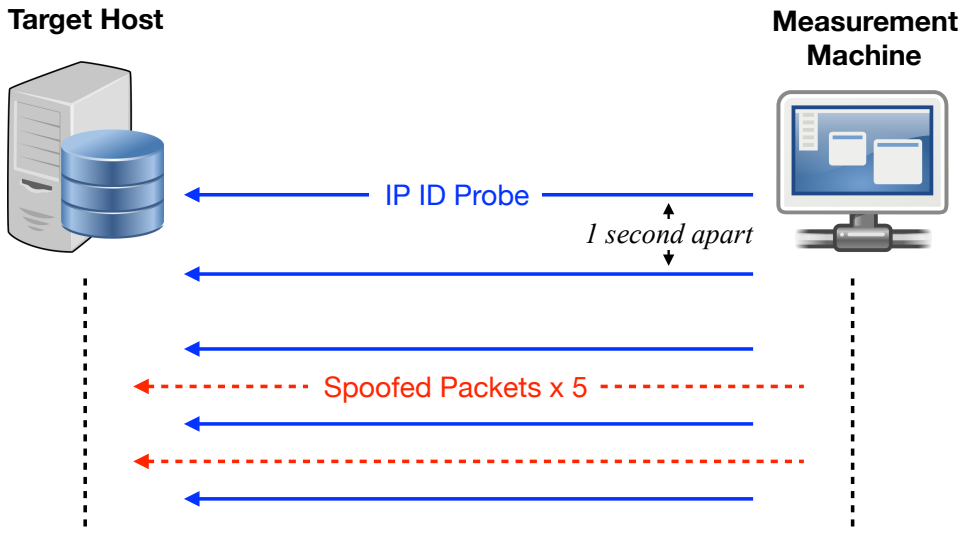


Figure 4.4. Blocking detection methodology.

If the reflector does not block the blacklist IP, then I will observe an IP ID increase sequence in the received RST responses as:

$$[\,+1,+1,+6,+6\,]$$

Here the last two deltas are +6 since the reflector does not block the blacklist IP and thus responds to spoofed packets, causing IP ID to increase by 5, and the probe packet causes it to increase by another 1, which together make +6.

On the other hand, if the reflector blocks the blacklist IP, then I will see an IP ID increase sequence as:

$$[\,+1,+1,+1,+1\,]$$

Here the last two deltas are +1 since the reflector blocks the blacklist IP, leading to no extra change in IP ID.

The first three probes — corresponding to the first two IP ID deltas — act as a control. The last two "probe and spoof" patterns perform the actual measurement. Seeing the initial

two “+1” indicates this host is in a quiet period — no extra network traffic. Therefore, I can be more confident that the following IP ID jump (“+6” in this case) is because of the experiment. However, while I present the choice of the numbers in the experiment as fait accompli, there is a rationale behind the choice of numbers which I discuss in Section 4.2.6.

Inference Criteria

Now I discuss how to infer whether a reflector is blocking a blacklist IP in the real world. I have limited vantage points from the measurement machine, as such, and my information is limited to the IP IDs I see from the reflector. Therefore, I would like to be very conservative when making a judgment. In this measurement, my approach is to try the same trial, between a reflector and a blacklist IP, many times until I get a “perfect signal” — a response which matches all the criteria below:

1. The measurement machine received exactly five RST responses from the reflector.
2. The five responses are received one second apart consecutively.
3. The IP ID increase sequence in the five responses are either [+1, +1, +6, +6], which I will conclude as no blocking, or [+1, +1, +1, +1], which I will conclude as blocking.
4. If any of the above three criteria are not met, I repeat the same experiment again. I repeat up to 15 times before giving up.

Essentially, the first requirement ensures no packet loss. The second requirement ensures responses I received reflect the real IP ID changes in the reflector. The Internet does not guarantee the order of packet arrival. Although I send one probe packet per second, and send the spoofed packets in between of the probe packets, these packets might not arrive at the reflector in the same order. There is a similar case for response packets. Therefore, the IP ID sequence I get from the response packets might not represent the real order of IP ID changes in the host. Requiring that the received response packets are also close to 1 second apart, I minimize the probability of

the reordered packets. In my experiment, I enforce that the response packets can not be less than 0.85 or more than 1.15 seconds apart.

The third requirement is the core of my inference logic. I want to be conservative when concluding whether there is blocking or not, so I will make the judgment only when I observe an IP ID increase sequence [+1, +1, +1, +1] or [+1, +1, +6, +6], and ignore everything else. Then if I saw a sequence of [+1, +1, +1, +1] but the host is not blocking the blacklist IP, that would mean all the 10 spoofed packets were lost during the transmit. On the other hand, if I see [+1, +1, +6, +6] and the host is actually blocking the blacklist IP, then that would mean during the experiment, there are exactly five extra packets generated by the host during each of the last two windows. Both cases are very unlikely to happen, and I will show a concrete analysis of false positives and false negatives in the remainder of this section.

False Positive and False Negative Analysis

For the experiment, a "false positive" is when a host is not blocking a blacklist IP, but I mistakenly conclude it is blocking. On the other hand, a "false negative" is when a host is blocking a blacklist IP, but I mistakenly conclude it is not blocking. With reflectors being collected, I can empirically evaluate the probability of a false positive or a false negative precisely.

For false positive evaluation, I first acquire a list of IPs that are verifiably not being blocked by reflectors. Since I own these IPs, I can easily verify that by directly probing reflectors from these IPs. I acquired and tested 1,265 IPs from five different /24s. Then I probe reflectors and send the spoofed packets with source addresses set to these pre-selected IPs. Since I know that these IPs are not blocked, if I observe an IP ID increase sequence of [+1, +1, +1, +1], then I know it is a false positive.

For false negative, I run the experiment with only probe packets, and no spoofed packets. This is equivalent to the scenario where the host blocks the spoofed IP. Then if I observe an IP ID increase sequence of [+1, +1, +6, +6], then I know it was due to the background traffic at the reflector and hence is a false negative.

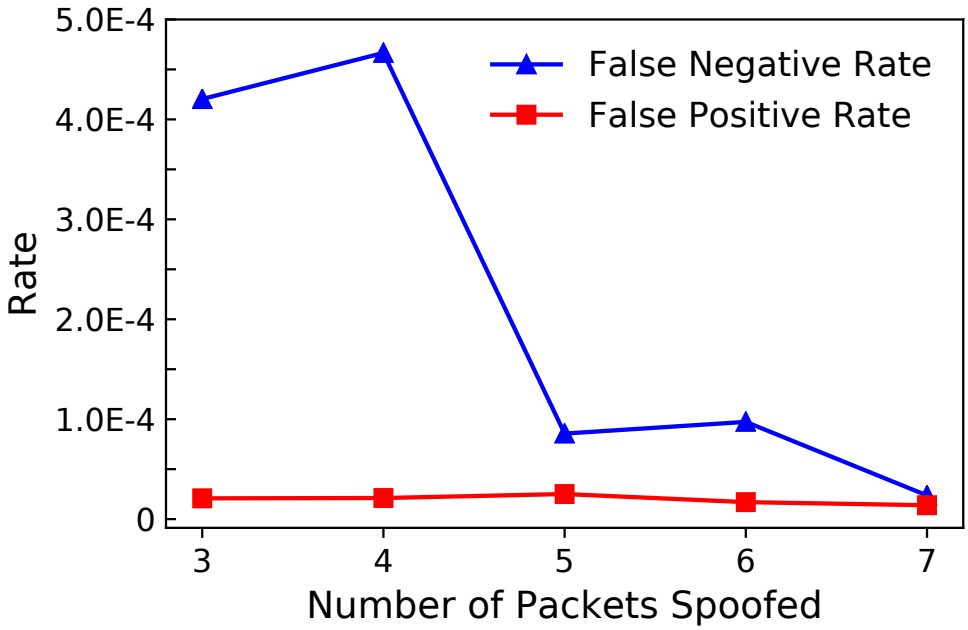


Figure 4.5. False positive rates and false negative rates of the technique when spoofing different amount of packets.

Although I have presented the design where I spoof five packets in each of the last two seconds, I also experimented with a range of numbers and calculated their false positive and negative rates. I tested with spoofed packets equal to 3, 4, 5, 6, 7 respectively. For each number, I use 15 distinct IPs I own as the source addresses to spoof, and I create another group with 15 place holder IPs where I do not send spoofed packets during the experiment. I run each experiment twice, and the final results are shown in Figure 4.5.

A few things stand out. The false negative rate drops significantly when I send 5 spoofed packets. Surprisingly, the false negative rate jumps up slightly when I spoof 6 packets. On the other hand, the false positive rate keeps marginally trending downwards as I increase the number of spoofed packets. I try to make the trade-off between having low false positive and negative rates, but at the same time generating as little traffic as possible. I choose 5 spoofed packets as a balance. By sending 5 spoofed packets, I get a false positive rate of 2.5e-5, and a false negative rate of 8.5e-5.

Furthermore, I also experimented with strategies where I send 4 probe packets, from which I get 3 IP ID deltas, and sending 6 probe packets, from which I get 5 IP ID deltas. With only 3 deltas I suffer a higher false negative rate, as it is easier for the reflector to show the same IP ID increase sequence with extra traffic. With 6 probes, on the other hand, I prolong the experiment, and more importantly, it is harder for us to get the "perfect signal" since it is harder to capture a period with no other traffic when the time window is longer. Thus, the choice of 5 probe packets with 5 spoofed packets in between is a trade-off between multiple factors.

Here we use a different mathematical model than the sequential hypothesis testing proposed in previous work [97]. We choose this simple and stringent method primarily because we try to be extremely conservative. Also, we like to keep each trial "short"—with only 5 seconds— to make the overall experiment faster, as we intend to use this technique for a large scale measurement. Using a statistical model could potentially cover more hosts, but there will not be a fundamental difference, since in either case we still rely on the fact that reflectors having a minimum amount of extra traffic.

4.2.7 Control Group

To further validate the measurements, every time I test a set of blacklist IPs against each reflector, I also include a control group of 20 randomly chosen IPs. These IPs are chosen from networks geo-located in the United States. I further ensure they do not appear on any of the blacklists, they are BGP routable, and they are not from the same ASes as the reflectors. The purpose of the control group is to create a random set of IPs that are unlikely to be blocked in bulk by a reflector. I use US IPs to avoid the potential problem of geo-blocking. If a reflector does block a significant fraction of control IPs, it is probably because the reflector is not suitable for this methodology (one reason can be that the ingress-filtering step did not catch these IPs). I discover 91 reflectors that constantly show blocking behavior for all control IPs, while the remaining reflectors never block more than 10 of the control IPs. I conclude that these 91 reflectors are not useful for measurement, and remove them from the total reflector set.

4.2.8 Ethical Considerations

In our experiments, we send spoofed packets to reflectors impersonating traffic from other IPs to infer the presence of network-layer blocking based on IP blacklists. A key ethical concern with this kind of measurement is the extent to which either receiving such packets or being seen to have received such packets would put the recipients at undue risk. Indeed, this is particularly problematic in censorship measurements [43, 97] because of the potential to inadvertently cause a host to be associated with content that is politically dangerous in their country. However, our work operates in a context that is substantially less risky, and we have further designed multiple aspects of our protocol to minimize the likelihood of risk. In particular, our methodology incorporates the following approaches to minimize risk:

Restriction in Scope: We have specifically restricted our measurements to only reflectors within the United States, which affords relatively robust free speech rights and considerable transparency around criminal proceedings. Indeed, from our conversations with both network operators and law enforcement, we are unaware of a realistic scenario where the mere receipt of a packet, regardless of its declared origin, has led to criminal or civil liability.

Conventional sources: Unlike in censorship studies, the source IP addresses being spoofed are those that have been used to mount wide-spread abusive activity such as spamming, port scanning, etc. and these represent precisely the kinds of traffic that a typical host on the Internet would *expect* to receive (This is the very theory behind using such blacklists). Thus, while the traffic we generate may appear malicious, it will be precisely the kind of malicious traffic we all experience via typical Internet background radiation [94].

Inbound, connection-free probes: Our measurements are constructed to be inbound only and connection free; that is, a network monitor could witness traffic consistent with an *external* scan of one of their hosts, but will never witness a completed connection or any data transmission. The only traffic ever sent *by* reflectors are RST packets. From our discussions with network operators and network security equipment vendors, we could not identify a scenario

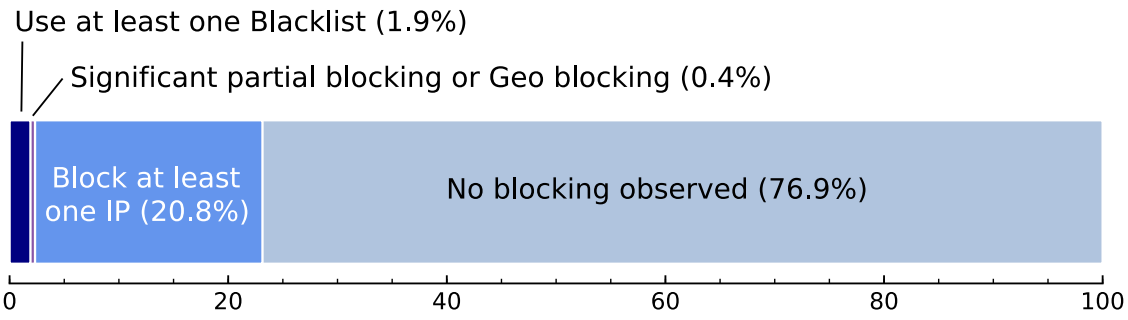


Figure 4.6. Breakdown of reflector blocking based on three experimental runs.

where the mere receipt of the packets we send would be sufficient to drive an incident response team to action.

Minimal use of end-host resources: Our scans are purposely constructed with SYN-ACK packets to ensure that no state is created on the reflector. Moreover, our peak probing rate per reflector is 6 min-sized packets per second, but even that rate only persists for two seconds in each test, and in the following pilot study, we probe each reflector no more than once every 3 minutes.

4.3 Overall Reflector Blocking

The methodology provides the basis for performing blacklist blocking measurements at scale: a measurement technique to confidently determine whether a reflector is blocking a particular IP address, a viable set of reflectors that are compatible with the technique, and a set of public security-related blacklists that provide a large set of candidate IPs that hosts might block. In this section, I describe my large-scale experiment that uses this methodology for determining which reflectors block IPs on the public blacklists, and which IPs they block. I then present the overall results of the blocking behavior of reflectors, and subsequent sections explore the different behaviors in more detail.

For a particular experimental run, I randomly selected 25 IPs from each blacklist that satisfies the requirements defined in Section 4.2: exclusive, stable, routable, geo-diversified,

and AS disjoint. Then I evaluated the blocking behavior for all 220K reflectors against the 225 blacklist IPs sampled from all nine blacklists. To increase the chances that these sampled IP will be stable, also to handle cases where reflectors might update the blacklists slowly(a reflector might take a long time to start blocking the newest IPs in a blacklist, although I discover later that it is not the case, see Section 4.4.2), I ensure the sampled IPs have stayed in the blacklist for at least 2 weeks before the experiment. Since for each blacklist, an experimental run can takes days to perform, as a post-processing step I remove blacklist IPs from consideration that did not remain on the blacklist for the duration of the experiment.

To increase the amount of evidence of blacklist blocking behavior, I conducted three experimental runs, each time using a different set of 25 IPs from each blacklist. I then conclude that a reflector is using a blacklist if only if all experiment runs show that it blocked all the sampled stable IPs from that blacklist.

I conducted the measurements from December 3–23rd, 2019. During this period, I tested 96,067,051 distinct `(reflector,IP)` pairs[3]. Based upon the criteria from Section 4.2, I are able to conclusively determine the blocking behavior (blocking or not blocking) of 98.3% of the tested pairs. Among these pairs, 894,570 pairs display a clear signal indicating "blocking".

Figure 4.6 presents the blocking behavior of all 222,782 reflectors I tested partitioned into four categories: those reflectors that I conclude use at least one of the public blacklists (1.9%), reflectors that block a large fraction of IPs on at least one blacklist in every experiment(0.4%, see more in Section 4.5), remaining reflectors that block at least one blacklist IP (20.8%), and reflectors that do not block any blacklist IPs (76.9%). (I identified 4,253 reflectors that use at least one blacklist (Section 4.4). I also discovered reflectors that block a significant fraction of blacklist IPs, due in part to geo-blocking (Section 4.5). Finally, I identified a large number of reflectors blocking at least one IP, suggesting wider use of a much larger set of blacklists (Section 4.6).) Note that given the requirements for hosts to be reflectors, such as running old

[3]The first two experiment I tested against all reflectors, the last experiment I only tested against the ones that have shown blocking behavior in the first two tests

Table 4.3. The number of reflectors using each of the nine different blacklists, as well as the number of unique /24s and ASes those reflectors appear in.

Blacklist (abbr.)	**Reflectors**	**/24s**	**ASes**
Spamhaus DROP (DROP)	4,142	1,782	50
Spamhaus EDROP (eDROP)	1,272	362	25
DShield Top Blacklist (DTop)	223	69	18
ET Compromised (ET)	116	58	15
BDS IP Ban List (BDS)	85	41	3
Feodo IP Blacklist (Feodo)	64	26	16
Snort IP Filter List (Snort)	52	20	11
Blocklist De Blacklist (DE)	36	18	8
Tor IP Blacklist (Tor)	24	9	8
Total Unique	4,253	1,827	77

OS versions, it is not surprising a large percentage shows no blocking of the blacklist IPs: they already have attributes anti-correlated with high degrees of security hygiene. Consequently, I want to emphasize that one should not conclude that this percentage is representative of all hosts on the Internet.

These high-level results provide the foundation for additional analyses and experiments, and going forward I further investigate each of these categories of reflector blocking behavior in turn. Section 4.4 explores blacklist use among the reflectors, Section 4.5 then examines significant partial blocking behavior (including geo-blocking), and Section 4.6 explores how reflectors that show any blocking behavior can be used as evidence of much broader use of blacklists. As a final analysis, Section 4.7 studies the consistency of reflector blocking behavior at a coarser granularity.

4.4 Reflectors Using Blacklists

In this section, I focus on the reflectors that use the blacklists I study, including the relative popularity of the blacklists, patterns in the use of multiple blacklists, and the rate at which reflectors update. I also use external sources of ground truth to validate my findings. Overall, I conclude from the results in this section that my methodology is indeed effective at

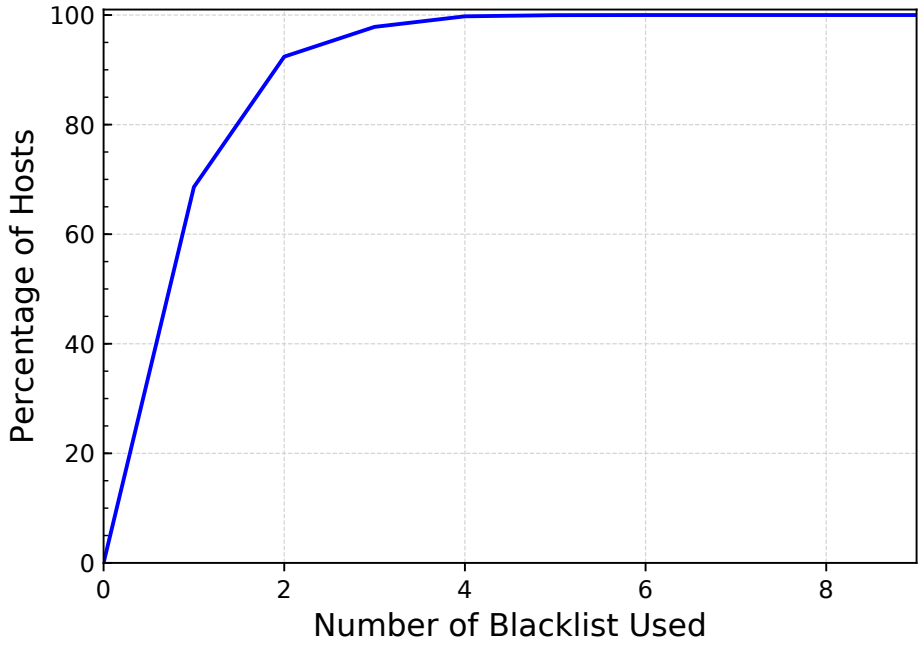


Figure 4.7. CDF of the number of blacklists used by reflectors

identifying blacklist use from a remote third-party vantage point.

Recall that I use three experimental runs that test whether reflectors block 25 randomly chosen IPs from the blacklists, and only conclude that a reflector uses a blacklist if it blocks *all* stable IPs on that blacklist across all runs. Based on these criteria, I identified 4,253 reflectors that use one of these public blacklists. Table 4.3 shows the number of reflectors using each of the nine different blacklists, as well as the number of unique /24s and ASes those reflectors appear in.

Spamhaus DROP is by far the most popular blacklist in the collection, followed by Spamhaus EDROP. The remaining blacklists have a comparatively small number of reflectors using them. On one hand, since many aspects of my methodology and experiment make conservative choices, these results should be considered a lower bound. On the other, one should have very high confidence in these results: I believe these reflectors are actually on networks that block the IPs on these blacklists.

4.4.1 Multiple Blacklist Use

For the reflectors using at least one blacklist, Figure 4.7 shows the cumulative distribution of the number of blacklists they use. At least for the most popular public blacklists I study, most use just one. Over 68.6% use just one blacklist, 23.8% use two or more, and only 7.6% use three or more. I find one reflector using six of the nine blacklists – the most I see in my study.

For these reflectors, though, there are interesting patterns to the multiple blacklists used. Figure 4.8 shows the use of multiple blacklists with a heatmap. Each cell shows the fraction of the reflectors using the blacklist in the row R that are also using the blacklist in the column C: $|R \cap C|/|R|$. Rows and columns correspond to blacklists, and each cell of the heatmap shows the fraction of the reflectors using the blacklist in row R that are also using the blacklist in column C. For example, the first cell for ET Compromised shows that 78% of the reflectors that use ET also use the Spamhaus DROP blacklist. Diagonal cells are 1.00 since they show blacklists compared with themselves.

The first cell of the Spamhaus EDROP row indicates that all reflectors that use Spamhaus EDROP also use Spamhaus DROP. Since the eDROP list is an extension of the DROP list, the behavior is strongly consistent with expectations (and, as such, is also a minor validation of the methodology). Moreover, the many significant values in the first two columns show that reflectors that use any of the other blacklists very often also use Spamhaus DROP and eDROP. These results underscore the popularity of Spamhaus DROP, and indicate that if a reflector blocks traffic using blacklists, it very likely uses Spamhaus DROP.

4.4.2 Reflector Update Latency

The objective of IP blacklists is to capture the most up-to-date IPs associated with malicious activities. As new threats come and go on the Internet, the content on the lists keeps changing. This dynamic nature of underlying threats and thus the IPs on the blacklists underscores the importance of reflectors keeping their blacklists recent. The goal of our next experiment is to estimate the delay between when new IPs appear on a blacklist and when reflectors start blocking

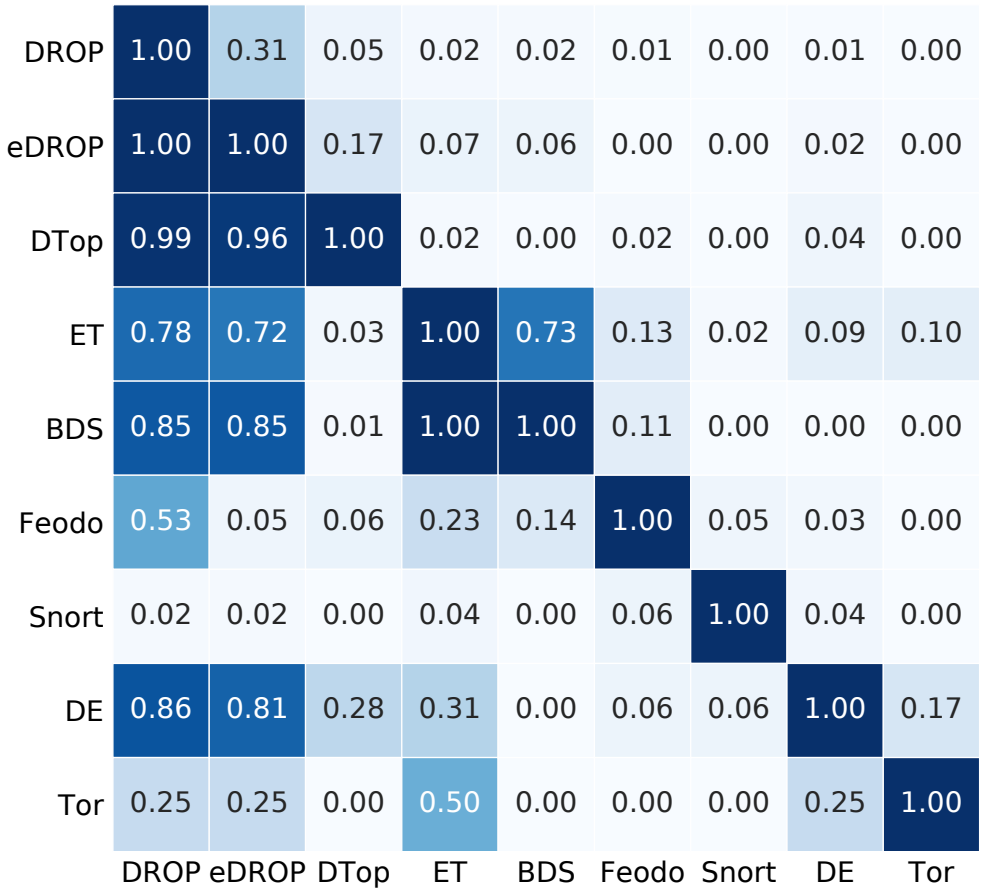


Figure 4.8. Pair-wise overlap of reflectors using the different blacklists.

them.

Because a single pass of the full set of reflectors takes hours, we examine the blacklist update behavior of reflectors at the granularity of a day. For each blacklist we identify newly added IPs between two consecutive days, and consider only those new IPs that were not on the blacklist in the prior two weeks. For this measurement, the blacklist IPs do not need to be exclusive, but still need to be routable. One day after the new IPs appear on a blacklist, we then test whether the reflectors using that blacklist now block these IPs, we repeat the experiment daily afterward if necessary. We also conduct this latency experiment twice to ensure consistent results(Since Tor IP Blacklist is a synthesized list, we did not measure the update latency regarding this blacklist).

Aside from one outlier, we find that *all* reflectors track updates to the blacklists. The reflectors that use the six non-Spamhaus blacklists all update within the one-day period. The outlier is a group of reflectors using Spamhaus DROP that stop updating after late October 2019 (We observed this by testing with the newly added IPs in Spamhaus DROP in the past), all the other reflectors that block Spamhaus DROP update within one-day period. After investigating, we found that all these outlier reflectors are located in one organization (a hosting provider). We suspect this organization stops updating their list after that October.

For Spamhaus EDROP, it hasn't added new IPs since Dec 17, 2019 as of Feb 14, 2020. We tested all the corresponding reflectors with the newly added IPs in Spamhaus EDROP back on Dec 17, 2019 and before, and found all the reflectors are update to date.

4.4.3 Validation

We are able to infer the use of blacklists for various hosts from our measurement. Ideally, we would also want to validate our findings. After checking the organizations where our reflectors are from, we reached out to 2 universities that we conclude are using blacklists. We got the ground truth regarding the exact blacklists they are using, and successfully validated our findings. More specifically, University *A* confirmed our findings that they use BDS IP Ban List, ET Compromised, Spamhaus DROP and Spamhaus EDROP. University *B* confirmed our finding that they use Spamhaus DROP and Spamhaus EDROP.

4.5 Partial Blocking

When performing the experimental runs I noticed that a small percentage of reflectors consistently blocked a significant subset of blacklist IPs, but not all, in *every experiment*. This consistency suggests that, while the reflector may not use the exact blacklist, there is a large overlap between the blacklist and the blocking policy of the reflector. We refer to such reflectors as exhibiting *significant partial blocking* behavior. Figure 4.6 shows these reflectors are just 0.4% of all reflectors that I tested, but they still correspond to 21% of the number of reflectors that

perfectly block at least one blacklist and therefore motivate further investigation. As a result, in this section I characterize this partial blocking behavior in more detail.

4.5.1 Geo-Blocking

Geo-blocking is one type of blocking I identified that contributes to this partial blocking. A reflector using geo-blocking will drop all traffic from a particular country. Organizations typically use geo-blocking either for policy reason (e.g., block GDPR countries [16]), or for security reasons (e.g., block countries that are sources of attacks, such as Russia or China). If a reflector uses geo-blocking, one will observe it blocking IPs on a blacklist if those IPs happen to be located in a blocked country. Although I take extra efforts to increase the geo diversity when sampling IPs(Section 4.2.5), this kind of overlap can still be exacerbated if a blacklist happens to have concentrations of IPs from particular countries. For example, DShield Top Blacklist on January 25, 2020 had over 50% of its IPs geo-located in the Netherlands. If a reflector blocks traffic from the Netherlands, then I would observe that the reflector is partially blocking DShield Top Blacklist.

To identify whether a reflector uses geo-blocking, I test whether the reflector consistently blocks a set of IPs from a particular country. For all countries related to blacklist IPs that I test, I first enumerate the IP address prefixes for those countries using four IP-based location services: MaxMind [84], IP2Location [60], IPDeny [61], and IPIP.net [62]. For each country, I then randomly select 20 IP addresses from those prefixes such that: (1) all four location services agree on the country label for the IPs, (2) the IPs do not appear on a blacklist, and (3) the IPs are BGP routable. Then for all reflectors, I test whether it blocks all of the randomly-chosen IPs for each country. If it does, then I conclude that it uses geo-blocking.

I tested the reflectors against 20 countries in total, ranging from large countries like Russia and China to small island countries like Singapore and Seychelles. Overall, only a small number of reflectors, 614 (0.28%), block at least one country, and 432 block at least two. China is blocked most often, with 501 of the 614 reflectors blocking random IPs in China. Russia is

Table 4.4. Number of reflectors exhibiting significant partial blocking on each blacklist.

	Blocked over 75%			Blocked over 50%		
Blacklist	**Hosts**	**/24s**	**ASes**	**Hosts**	**/24s**	**ASes**
DROP	28	18	3	23	21	3
eDROP	96	60	32	49	27	18
DTop	157	66	19	319	165	72
ET	13	7	6	31	19	17
BDS	8	5	5	7	7	7
Feodo	65	30	19	23	17	15
Snort	11	9	7	34	20	17
DE	148	38	1	13	11	4
Tor	63	35	26	31	19	16
Total	492	207	71	459	257	108

second at 376, followed by Hong Kong (177) and Vietnam (175). European countries including Belgium, Netherlands, and France also have over 60 reflectors blocking them.

Note that the methodology identifies geo-blocking at the network layer. Other forms of geo-blocking exist, such as application-layer blocking (e.g., HTTP 403 Forbidden). I do not explore all possible geo-blocking mechanisms since my goal was to identify reflectors using geo-blocking at the network layer.

4.5.2 Significant Partial Blocking

In addition to geo-blocking, there are reasons why a reflector may block a blacklist IP that is not due to the reflector using that blacklist. A host may have internal policies that deny access from some network providers, or network administrators may add IPs into their firewall on an ad-hoc basis based on an organization's internal strategies or policies. These alternate blocking behaviors could overlap with the blacklist IPs I sampled, leading to partial blocking behavior in the results.

Having identified reflectors using geo-blocking, I remove these reflectors from further consideration. I then calculate two groups of reflectors: for each blacklist, I identify reflectors that partially block over 75% of the sampled IPs *every time* I test them, and another group where

they partially block over 50% of the sampled IPs.

For each blacklist, Table 4.4 summarizes the number of reflectors that fall into each category. If a reflector is blocking more than 75% of the sampled IPs every time, it is plausible that the reflector is using a source that is very similar to the blacklist I tested. For instance, there could be other blacklists where the data collection methodology is similar to the method used by the public blacklists in this study. Previous work has shown that commercial products can aggregate data from public blacklists, and that they potentially conduct post-processing to eliminate some content [77].

I suspect that the partial blocking behavior in Table 4.4 likely results from such cases. In particular, the number of reflectors that are partial blocking DShield Top Blacklist is relatively high compared with other blacklists—it covers over 30% of all reflectors in the first group and over 69% in the second group, suggesting that this list may be relatively frequently included into other lists.

4.6 Beyond The 9 Blacklists

Previously I chose blacklist IPs that were exclusive to each blacklist, effectively creating a set of IPs that served as a signature for that blacklist. But as one can see from Figure 4.6, there are over 20% of reflectors that blocked at least one blacklist IP during one of the three experiments, and I do not have a clear explanation. One possible case is that the blacklist IPs I sampled happen to overlap with some commercial blacklists we do not know, and those lists are more widely used. To check this possibility, I change the blacklist IP sampling criteria and conduct a new experiment.

Now my goal is the opposite: I want to choose among common, or shared, IPs that appear on multiple blacklists. As added caution, I also focus on shared IPs that are the most stable, appearing on the blacklists for at least three weeks. The goal is to bias selecting blacklist IPs that are so egregiously suspicious that not only do they appear on multiple of the public blacklists for a long period of time, but as a result they also likely appear on other public and commercial

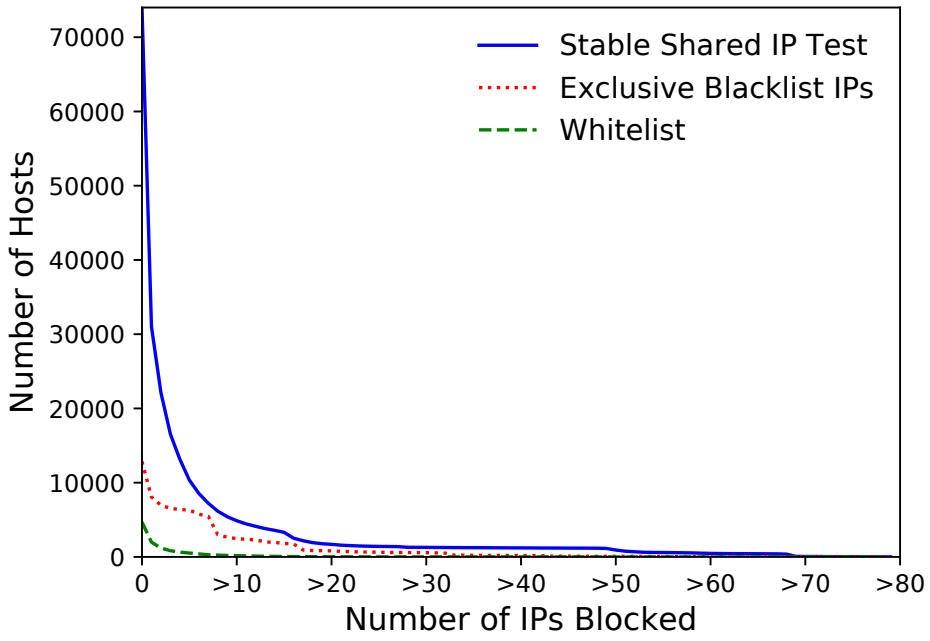


Figure 4.9. Number of reflectors (*y*-axis) that block at least some number of blacklist IPs (*x*-axis)

blacklists that I do not have access to.

For this experiment I randomly selected 80 blacklist IPs that appear on more than one blacklist and, as with previous experiments, that are routable and AS disjoint. I refer to this set as the "Shared" blacklist IPs. For comparison, I also randomly selected a set of 80 blacklist IPs from the previous experiment, the ones that are exclusive to one blacklist ("Exclusive"), and a random set of 80 IPs as a whitelist set ("Whitelist"), again with the routable and AS disjoint criteria. Here the whitelist is constructed by randomly sampling IPs from top 10,000 most visited IPs among all the network traffic in the own organization in one day (an education institute of over 30K students and faculty).

For each of these sets of 80 blacklist IPs, Figure 4.9 shows how many reflectors block the same number of IPs as a distribution over the number of blacklist IPs blocked. (I evaluated multiple random sets of shared blacklist IPs to see whether the random selections introduced noticeable variance. Since the results were very consistent across the different random sets, for

Table 4.5. Blocking behavior of reflectors with the top 12 most-blocked blacklist IPs.

	Granularity		
Blacklist IP	**IP**	**/24**	**AS**
178.73.215.171	25,749	175	210
185.176.27.98	15,560	5,598	1,690
185.175.93.103	13,550	1,155	1,224
92.63.194.115	11,889	2,028	1,442
176.10.99.200	9,049	466	238
185.156.73.54	8,863	1,623	5,441
80.67.223.41	6,689	624	624
176.100.109.3	4,871	553	610
171.25.193.25	4,468	212	148
216.239.90.19	4,421	56	39
185.220.102.8	4,297	273	273
62.210.37.82	4,058	528	300

clarity I just show results for one of them.) For example, for "Exclusive" blacklist IPs only 2,649 reflectors block 10 or more IPs. In contrast, for "Shared" blacklist IPs that appear on more than one blacklist, far more reflectors block them. Over 73K reflectors block at least one IP from the shared set, and 5,412 reflectors block 10 or more IPs. In contrast, for the whitelist set, there are only 202 reflectors that block 10 or more whitelist IPs(I checked these whitelist IPs and found the most blocked ones are from Cloudflare, which is a popular CDN network but known to associate with malicious activities). These results suggest that the number of Internet hosts that potentially use security-related traffic blocking is much larger than just the ones that use the public blacklists I study.

Of course, it is possible that this more extensive blocking behavior is not a result of security-related blocking, but rather because of other blocking policies such as geo-blocking. One notable difference, though, is that security-based traffic blocking usually targets individual IPs, whereas other policy-driven blocking can target a specific network or subnet. As another experiment, I check whether the blocking I observed indeed targets individual IPs, or instead larger network blocks. I select the top 12 IPs where they have the most amount of reflectors blocking them in the previous test. For each of these blacklist IPs, I randomly sample another

Table 4.6. The blocking consistency of the reflectors in /24 prefixes with more than one reflector.

	Consistent		Almost Consistent	Inconsistent
Blacklist	**No Blocking (%)**	**Consistent (%)**	**Off By One (%)**	**Inconsistent (%)**
BDS IP Ban List	35,540 (88.19%)	398 (0.99%)	2,332 (5.79%)	2,030 (5.03%)
Blocklist De Blacklist	40,228 (99.82%)	24 (0.06%)	13 (0.03%)	35 (0.09%)
DShield Top Blacklist	39,603 (98.27%)	393 (0.98%)	138 (0.34%)	166 (0.41%)
ET Compromised	39,535 (98.10%)	298 (0.74%)	102 (0.25%)	365 (0.91%)
Feodo IP Blacklist	39,943 (99.11%)	195 (0.48%)	42 (0.10%)	120 (0.31%)
Snort IP Filter List	39,830 (98.83%)	250 (0.62%)	88 (0.22%)	132 (0.33%)
Spamhaus DROP	39,320 (97.57%)	750 (1.86%)	30 (0.07%)	200 (0.50%)
Spamhaus EDROP	39,792 (98.73%)	349 (0.87%)	36 (0.09%)	123 (0.31%)
Tor IP Blacklist	40,000 (99.25%)	165 (0.41%)	43 (0.11%)	92 (0.23%)
Overall	353,791 (97.54%)	2,822 (0.78%)	2,824 (0.78%)	3,263 (0.90%)
Control Group	40,255 (99.89%)	3 (0.01%)	37 (0.09%)	5 (0.01%)

three IPs from the same /24, and randomly sample another four IPs from the same AS. For all of these IPs, I then measure how many reflectors block each of them.

Table 4.5 shows the number of reflectors that block the top 12 blacklist IPs, the random IPs from the same /24 as the blacklist IPs, and the random IPs from the same AS (The second column shows the number of reflectors that block the particular IP, the third column shows the *maximum* number of reflectors that block IPs from the same /24, and the last column shows the maximum number of reflectors that block IPs sampled from the same AS.). These results confirm that reflectors exhibit blocking behavior targeting specific IPs rather than larger network blocks. Indeed, searching the Web based on these blacklist IPs returns reports linking these IPs to a range of malicious activities, including massive port scanning, brute-force login attempts, and sending spam. That said, although I do not know the exact feeds these hosts are using, the results suggest that security-related network blocking is prevalent even among hosts such as these reflectors.

4.7 Blocking Consistency

As a final analysis, I explore the consistency of reflector blocking behavior at a coarser granularity. A common use case of blacklists is at the granularity of an organization, often via some kind of network appliance. In such a scenario, I would expect the blocking behavior of

reflectors to be consistent across an organization: if one reflector blocks a blacklist IP, then other reflectors in the same organization should also block it.

Ideally I would like to map reflectors to organizations to answer this question. However, mapping an IP to an organization is a challenging problem, particularly with the increasing use of WHOIS anonymization. Instead, I use the common, more tractable technique of aggregating reflectors by their /24 prefix. As a result, in this section I answer a methodological question: If I aggregate reflectors by their /24 prefix, do the aggregated reflectors exhibit consistent blocking behavior? Is the /24 prefix aggregation a useful proxy for expected consistent blocking by organizations?

My data set has 134,370 reflectors that are part of /24s with more than one reflector. For each blacklist, I categorize the blocking behavior of multiple reflectors in the same /24 into one of three categories: *consistent*, *almost consistent*, and *inconsistent*. I define a /24 to be "consistent" for a blacklist if *all* the individual reflectors in that /24 block the *exact same* blacklist IPs. A /24 is "almost consistent" if the blocking behavior of the reflectors in a /24 differs only by one IP. For example, a /24 is "almost consistent" if it has four reflectors, three of which block the same 21 IPs from a blacklist, and the fourth reflector blocks 20 out the same 21 IPs. Finally, I consider all other /24s "inconsistent".

Using these definitions, Table 4.6 shows the consistency results for all the /24s that have more than one reflector. The results are dominated by /24s that do not show any blocking behavior. I consider such /24s consistent since all the hosts under these /24s block the same number (zero) of blacklist IPs, but these cases also do not provide much insight.

Excluding the "no blocking" cases, then in the presence of any blocking, consistency of blocking behavior at a /24 granularity is far from guaranteed. As discussed in Section 4.2.6, the measurement technique has very low false positive and false negative rates. Measurement error can potentially explain some "almost consistent" cases and perhaps some "inconsistent" cases, too. However, the consistency results for the control group, comprised of 20 randomly sampled US IPs (Section 4.3), shows that the potential effect of measurement error on consistency is

small. In other words, the inconsistent cases do indeed demonstrate different blocking behavior among hosts within the same /24.

One situation that could lead to inconsistent blocking behavior within a /24 is when the network belongs to a cloud or hosting provider, and the IPs within the same /24 are used by distinct entities. For instance, when manually examining the inconsistent /24s for BDS IP Ban List (which has the highest inconsistency), I found more than 60% of these /24s belong to cloud or hosting providers. Another situation leading to inconsistent block behavior is when a /24 belongs to an ISP which sub-allocates IP addresses to different customers.

In summary, the results indicate that one can not assume consistent blocking behavior for reflectors in the same /24 network.

4.8 Discussion

In this chapter, I implement and test a technique for inferring the deployment of network layer blacklisting and, further, for attributing the use of particular blacklists in particular networks. There are a range of limitations in this pilot study, most significantly including potential selection bias arising from using quiescent U.S. hosts running older versions of Windows. This is a limitation of the methodology itself. Another limitation lies in the fact that I exclusive use public blacklist data (i.e., since I do not have access to high-priced commercial threat intelligence feeds which could be distinct).

However, even given these limitations, the measurements of 220K hosts reveal a number of interesting artifacts. First, I discovered the widespread use of *some* kind of network layer blocking (affecting over a third of hosts in the data set) even if it is not consistent with membership in any of the lists I track. This demonstrated the prevalence of security related blocking even among machines with low security hygiene. Most most previous network disruption measurement explain the reason as censorship (whether nation wide or at corporate level). My work highlights the prevalence of security related blocking, which serves as a reminder that for all the following network connectivity study, researchers should always keep in mind the possibility of security

related blocking.

Second, I find that there is evidence of intra-network diversity in traffic blocking policy. While a number of network prefixes have consistent blocking behavior across multiple hosts, quite a few do not, suggesting different network security policies are being employed on different subnets. This implies that when measuring network behaviors, researchers should not assume the consistency within a network.

Finally, for blacklist use that can be precisely attributed the most widely used blacklists (Spamhaus DROP and eDROP and DShield Top) are also those that have extremely low false positives. [4] This suggests that for many networks proactive traffic blocking is gated on having lists of sufficient accuracy to remove the risks of accidentally blocking legitimate traffic.

4.9 Acknowledgement

This chapter, in part, is a reprint of the material as submitted to the Proceedings of the Usenix Security 2020. *Clairvoyance: Inferring Blacklist Use on the Internet* Vector Guo Li, Gautam Akiwate, Yihui Chen, Geoffrey M. Voelker, Kirill Levchenko, Stefan Savage. The dissertation author was the primary investigator and author of this paper. I really appreciate the help from my coauthors, especially Gautam Akiwate. He helped me a lot on data collection and analysis, and I really learned a lot from the way he looks at data problems. Furthermore, he gave me crucial moral support during some critical times of the project. I will not be able to finish this work without his help. I also really appreciate the suggestions I received from kc Claffy and Alex Gantman regarding this project.

[4] The DROP and eDROP lists are a small subset of Spamhaus' feed that specifically deals with address for which the entire network prefix is believed to be abusive (e.g., prefix hijacking).

Chapter 5

Conclusion

This dissertation focuses on using empirical approach to study problems surrounding threat intelligence. In Chapter 3, I proposed a set of simple but also fundamental metrics, and measured a broad set of threat intelligence sources, and reported the characteristics and limitations of threat intelligence data. In Chapter 4, I designed a method and conducted a large scale measurement on how online hosts are using threat intelligence data. To summarize, I will first highlight the lessons I learned from my studies, and then discuss the takeaways of my study for the community. Here are the high-level lessons from my threat intelligence study:

- Threat intelligence feeds, far from containing homogeneous samples of some underlying truth, vary tremendously in the kinds of data they capture based on the particularities of their collection approach. Unfortunately, few threat intelligence vendors explain the mechanism and methodology by which their data are collected and thus threat intelligence consumers must make do with simple labels such as “scan” or “botnet”, coupled with inferences about the likely mode of collection. Worse, a significant amount of data does not even have a clear definition of category, and is only labelled as “malicious” or “suspicious”, leaving the ambiguity to consumers to decide what action should be taken based on the data.

- There is little evidence that larger feeds contain better data, or even that there are crisp quality distinctions between feeds across different categories or metrics (i.e., that a threat

intelligence provider whose feed performs well on one metric will perform well on another, or that these rankings will hold across threat categories). How data is collected also does not necessarily imply the feeds' attributes. For example, crowdsourcing-based feeds (e.g., Badips feeds), are not always slower in reporting data than the self-collecting feeds (like Paid IP Reputation).

- Most IP-based threat intelligence data sources are collections of singletons (i.e., that each IP address appears in at most one source) and even the higher-correlating data sources frequently have intersection rates of only 10%. Moreover, when comparing with broad sensor data in known categories with broad effect (e.g., random scanning) fewer than 2% of observed scanner addresses appear in most of the data sources I analyzed; indeed, even when focused on the largest and most prolific scanners, coverage is still limited to 10%. There are similar results for file hash-based sources with little overlap among them.

- Security related network blocking is relative prevalent on the Internet. The measurement has shown that many online hosts, even with our biased selection of low security hygiene hosts, are blocking network traffic based on threat intelligence data, or part of the data. When studying network connectivity disruption, researchers should not ignore the effect of these security related blocking, instead, we should model these behavior when conducting measurements and analyzing the results.

- The network behaviors within a subnet are not always the same, different hosts in the same network can have different behavior, and these cases are not neglectable. Researchers should not make such assumptions when measuring the behaviors of a network, and one should always be careful when trying to conclude a behavior with only a few vantage points in a network.

The low intersection and coverage of threat intelligence feeds could be the result of several non-exclusive possibilities. First is that the underlying space of indicators (both IP

addresses and malicious file hashes) is large and each individual data source can at best sample a small fraction thereof. It is almost certain that this is true to some extent. Second, different collection methodologies—even for the same threat category—will select for different sub distributions of the underlying ground truth data. Third, this last effect is likely exacerbated by the fact that not all threats are experienced uniformly across the Internet and, thus, different methodologies will skew to either favor or disfavor targeted attacks.

There are many ways people can use threat intelligence data. It can be used to *enrich* other information (e.g., for investigating potential explanations of a security incident), as a probabilistic canary (i.e., identifying an overall site vulnerability via a single matching indicator may have value even if other attacks of the same kind are not detected) or in providing a useful source of ground truth data for supervised machine learning systems. However, even given such diverse purposes, organizations still need some way to prioritize which threat intelligence products to invest in. The metrics I proposed in Chapter 3 provide some direction for such choices. For example, an analyst who expects to use threat intelligence interactively during incident response would be better served by feeds with higher coverage, but can accommodate poor accuracy, while an organization trying to automatically label malicious instances for training purposes (e.g., brute force attacks) will be better served by the converse.

Based on my experience analyzing threat intelligence data and how people are using it, I try to provide several recommendations for the security community on this topic moving forward:

- The threat intelligence community should standardize data labeling, with a clear definition of what the data means and how the data is collected. Security experts can then assess whether the data fit their need and the type of action should be taken on this data.

- There are few rules of thumb in selecting among threat intelligence feeds, as there is not a clear correlation between different feed properties. Consumers need empirical metrics, such as those I describe, to meaningfully differentiate data sources, and to prioritize certain

metrics based on their specific need.

- Blindly using threat intelligence data—even if one could afford to acquire many such sources—is unlikely to provide better coverage and is also prone to collateral damage caused by false positives. Customers need to be always aware of these issues when deciding what action should be taken on this data.

- Besides focusing on the threat intelligence data itself, future work should investigate the operational uses of threat intelligence in industry, as the true value of threat intelligence data can only be understood in operational scenarios. Moreover, the community should explore more potential ways of using the data, which will extend people's understanding of threat intelligence and also influence how vendors are curating the data and providing the services.

- Follow up on the previous point, it is also important to for the community to study the potential impact of people using threat intelligence. Since this is security related data, certain use cases, like blocking network traffic (either IP or DNS), could have broader impact on the Internet. As threat intelligence getting more and more popular, researchers should closely follow the impact of these products, so we can spot problems early on and make changes before heavy damage is made.

Bibliography

[1] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 60–69, 2007.

[2] Abuse.ch. https://abuse.ch/.

[3] Sadia Afroz, Michael Carl Tschantz, Shaarif Sajid, Shoaib Asif Qazi, Mobin Javed, and Vern Paxson. Exploring server-side blocking of regions. *arXiv preprint arXiv:1805.11606*, 2018.

[4] Top Alexa domains. https://www.alexa.com/topsites/.

[5] Alienvault IP reputation. http://reputation.alienvault.com/reputation.data.

[6] AlienVault Open Threat Exchange. https://otx.alienvault.com/.

[7] Daniel Anderson. Splinternet behind the great firewall of china. *Queue*, 10(11):40–49, 2012.

[8] Ion Androutsopoulos, John Koutsias, Konstantinos V Chandrinos, George Paliouras, and Constantine D Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *arXiv preprint cs/0006013*, 2000.

[9] Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Georgios Sakkis, Constantine D Spyropoulos, and Panagiotis Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. *arXiv preprint cs/0009009*, 2000.

[10] Anomali ThreatStream Threat Intelligence. https://www.anomali.com/products.

[11] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, and Michalis Kallitsis. Understanding the mirai botnet. In *USENIX Security Symposium*, 2017.

[12] Simurgh Aryan, Homa Aryan, and J Alex Halderman. Internet censorship in iran: A first look. In *Presented as part of the 3rd {USENIX} Workshop on Free and Open Communications on the Internet*, 2013.

[13] Badips. https://www.badips.com/.

[14] Michael Bailey, Jon Oberheide, Jon Andersen, Z Morley Mao, Farnam Jahanian, and Jose Nazario. Automated classification and analysis of internet malware. In *International Workshop on Recent Advances in Intrusion Detection*, pages 178–197. Springer, 2007.

[15] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. Scalable, behavior-based malware clustering. In *NDSS*, volume 9, pages 8–11. Citeseer, 2009.

[16] BBC News. GDPR: US news sites unavailable to EU users under new rules. https://www.bbc.com/news/world-europe-44248448, May 2018.

[17] Steven M Bellovin. A technique for counting natted hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 267–272, 2002.

[18] Karyn Benson, Alberto Dainotti, Alex C Snoeren, and Michael Kallitsis. Leveraging internet background radiation for opportunistic network analysis. In *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015.

[19] The Bro Network Security Monitor. https://www.bro.org/index.html.

[20] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Thomas Fischer, Ahmad-Reza Sadeghi, and Bhargava Shastry. Towards taming privilege-escalation attacks on android. In *NDSS*, volume 17, page 19. Citeseer, 2012.

[21] J. Butler and G. Hoglund. Vice–catch the hookers! *In Black Hat USA, July 2004*, 2004.

[22] Xavier Carreras and Lluis Marquez. Boosting trees for anti-spam email filtering. *arXiv preprint cs/0109015*, 2001.

[23] Composite Blocking List. https://www.abuseat.org/.

[24] Spreading the disease and selling the cure. https://krebsonsecurity.com/2015/01/spreading-the-disease-and-selling-the-cure/.

[25] Censys | Public Internet Search Engine. https://censys.io/.

[26] Neha Chachra, Damon McCoy, Stefan Savage, and Geoffrey M. Voelker. Empirically Characterizing Domain Abuse and the Revenue Impact of Blacklisting. In *Proceedings of the Workshop on the Economics of Information Security (WEIS)*, State College, PA, 2014.

[27] Madhusudhanan Chandrasekaran, Krishnan Narayanan, and Shambhu Upadhyaya. Phishing email detection based on structural properties. In *NYS cyber security conference*, volume 3. Albany, New York, 2006.

[28] Cisco Talos Threat Intelligence. https://www.cisco.com/c/en/us/products/security/talos.html.

[29] Cisco Next-Generation Firewall. https://www.cisco.com/c/en/us/products/security/firewalls/index.html.

[30] Richard Clayton, Steven J Murdoch, and Robert NM Watson. Ignoring the great firewall of china. In *International Workshop on Privacy Enhancing Technologies*, pages 20–35. Springer, 2006.

[31] Cloudflare, fast, global content delivery network. https://www.cloudflare.com/cdn/.

[32] AWS CloudFront, fast, highly secure and programmable content delivery network. https://aws.amazon.com/cloudfront/.

[33] William W Cohen. Learning rules that classify e-mail. In *AAAI spring symposium on machine learning in information access*, volume 18, page 25. Stanford, CA, 1996.

[34] Cyber Threat Alliance. https://www.cyberthreatalliance.org/.

[35] CrowdStrike Cyber Threat Intelligence Platform Falcon X. https://www.crowdstrike.com/endpoint-security-products/falcon-x-threat-intelligence/.

[36] Cyber Observable eXpression. http://cyboxproject.github.io/documentation/.

[37] Louis F DeKoven, Stefan Savage, Geoffrey M Voelker, and Nektarios Leontiadis. Malicious browser extensions at scale: Bridging the observability gap between web site and browser. In *10th USENIX Workshop on Cyber Security Experimentation and Test (CSET 17)*. USENIX, 2017.

[38] Harris Drucker, Donghui Wu, and Vladimir N Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural networks*, 10(5):1048–1054, 1999.

[39] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1285–1298, 2017.

[40] Thomas Dullien and Rolf Rolles. Graph-based comparison of executable objects (english version). *SSTIC*, 5(1):3, 2005.

[41] Zakir Durumeric, Michael Bailey, and J Alex Halderman. An internet-wide view of internet-wide scanning. In *USENIX Security Symposium*, 2014.

[42] Edgecast CDN, Verizon digital and media services. https://www.verizondigitalmedia.com/platform/edgecast-cdn/.

[43] Roya Ensafi, Jeffrey Knockel, Geoffrey Alexander, and Jedidiah R Crandall. Detecting intentional packet drops on the internet via tcp/ip side channels. In *International Conference on Passive and Active Network Measurement*, pages 109–118. Springer, 2014.

[44] Facebook threat exchange. https://developers.facebook.com/programs/threatexchange.

[45] Facebook ThreatExchange. https://developers.facebook.com/programs/threatexchange/.

[46] Fastly managed CDN. https://www.fastly.com/products/fastly-managed-cdn.

[47] Paul Ferguson and Daniel Senie. rfc2827: network ingress filtering: defeating denial of service attacks which employ ip source address spoofing, 2000.

[48] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th international conference on World Wide Web*, pages 649–656, 2007.

[49] FireHOL IP Lists | All Cybercrime IP Feeds. http://iplists.firehol.org/.

[50] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji, and Thomas A Longstaff. A sense of self for unix processes. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 120–128. IEEE, 1996.

[51] Fortinet FortiGuard Threat Intelligence. https://www.fortinet.com/fortiguard/threat-intelligence/threat-research.html.

[52] Fortinet Next-Generation Firewall. https://www.fortinet.com/products/next-generation-firewall.html.

[53] Marius Gheorghescu. An automated virus classification system. 2006.

[54] Ashvin Goel, Kenneth Po, Kamran Farhadi, Zheng Li, and Eyal De Lara. The taser intrusion recovery system. In *Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 163–176, 2005.

[55] Shuang Hao, Alex Kantchelian, Brad Miller, Vern Paxson, and Nick Feamster. Predator: proactive recognition and elimination of domain abuse at time-of-registration. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1568–1579. ACM, 2016.

[56] IBM X-Force Exchange. https://exchange.xforce.ibmcloud.com/.

[57] Information controls lab. https://iclab.org/.

[58] OpenNet Initiative. Survey of government internet filtering practices indicates increasing internet censorship, may 2007.

[59] Incident Object Description Exchange Format. https://tools.ietf.org/html/rfc5070.

[60] IP2Location: IP Address to Identify Geolocation Information. https://www.ip2location.com/.

[61] IPdeny IP country blocks. https://www.ipdeny.com/.

[62] IPIP.net: The only IP Database based on real time BGP/ASN data analytics. https://en.ipip.net/.

[63] Nav Jagpal, Eric Dingle, Jean-Philippe Gravel, Panayiotis Mavrommatis, Niels Provos, Moheeb Abu Rajab, and Kurt Thomas. Trends and lessons from three years fighting malicious extensions. In *USENIX Security Symposium*, 2015.

[64] Jaeyeon Jung and Emil Sit. An empirical study of spam traffic and the use of dns black lists. In *Proceedings of the ACM Conference on Internet Measurement*, 2004.

[65] Alexandros Kapravelos, Chris Grier, Neha Chachra, Christopher Kruegel, Giovanni Vigna, and Vern Paxson. Hulk: Eliciting malicious behavior in browser extensions. In *USENIX Security Symposium*. San Diego, CA, 2014.

[66] Sheharbano Khattak, David Fifield, Sadia Afroz, Mobin Javed, Srikanth Sundaresan, Vern Paxson, Steven J Murdoch, and Damon McCoy. Do you see what i see? differential treatment of anonymous users. Internet Society, 2016.

[67] Samuel T King and Peter M Chen. Backtracking intrusions. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 223–236, 2003.

[68] Samuel T King, Zhuoqing Morley Mao, Dominic G Lucchetti, and Peter M Chen. Enriching intrusion alerts through multi-host causality. In *NDSS*, 2005.

[69] Engin Kirda, Christopher Kruegel, Greg Banks, Giovanni Vigna, and Richard Kemmerer. Behavior-based spyware detection. In *Usenix Security Symposium*, page 694, 2006.

[70] Amit Klein and Benny Pinkas. From IP ID to device id and KASLR bypass. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1063–1080, 2019.

[71] Marc Kührer, Christian Rossow, and Thorsten Holz. Paint it black: Evaluating the effectiveness of malware blacklists. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2014.

[72] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. High accuracy attack provenance via binary-based execution partition. In *NDSS*, 2013.

[73] Tony Lee. Behavioral classification. *Proceedings of EICAR 2006, 4*, 2006.

[74] Wenke Lee and Salvatore Stolfo. Data mining approaches for intrusion detection. 1998.

[75] Jonathan Lemon. Resisting syn flood dos attacks with a syn cache. In *BSDCon*, volume 2002, pages 89–97, 2002.

[76] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Márk Félegyházi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, Damon McCoy, Nicholas Weaver, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage. Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *Proceedings of the IEEE Symposium and Security and Privacy*, 2011.

[77] Vector Guo Li, Matthew Dunn, Paul Pearce, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. Reading the tea leaves: A comparative analysis of threat intelligence. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 851–867, Santa Clara, CA, August 2019. USENIX Association.

[78] Ulf Lindqvist and Phillip A Porras. Detecting computer and network misuse through the production-based expert system toolset (p-best). In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344)*, pages 146–161. IEEE, 1999.

[79] Yushan Liu, Mu Zhang, Ding Li, Kangkook Jee, Zhichun Li, Zhenyu Wu, Junghwan Rhee, and Prateek Mittal. Towards a timely causality analysis for enterprise security. In *NDSS*, 2018.

[80] Ponemon Institute LLC. Third annual study on changing cyber threat intelligence: There has to be a better way. January 2018.

[81] LogRhythm Security Information and Event Management. https://logrhythm.com/solutions/security/siem/.

[82] Shiqing Ma, Xiangyu Zhang, and Dongyan Xu. Protracer: Towards practical provenance tracing by alternating between logging and tainting. In *NDSS*, 2016.

[83] MaxCDN. https://www.maxcdn.com/one/.

[84] MaxMind: IP Geolocation and Online Fraud Prevention. https://www.maxmind.com/en/home.

[85] Allison McDonald, Matthew Bernhard, Luke Valenta, Benjamin VanderSloot, Will Scott, Nick Sullivan, J Alex Halderman, and Roya Ensafi. 403 forbidden: A global view of cdn geoblocking. In *Proceedings of the Internet Measurement Conference 2018*, pages 218–230, 2018.

[86] Leigh Metcalf and Jonathan M Spring. Blacklist ecosystem analysis: Spanning jan 2012 to jun 2014. In *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*. ACM, 2015.

[87] Darren Mutz, Fredrik Valeur, Giovanni Vigna, and Christopher Kruegel. Anomalous system call detection. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):61–93, 2006.

[88] NetAcuity Industry-Standard Geolocation. https://www.digitalelement.com/solutions/.

[89] Nothink honeypot SSH. http://www.nothink.org/honeypot_ssh.php.

[90] OONI - Open Observatory of Network Interference. https://ooni.org/.

[91] Packetmail.net. https://www.packetmail.net/.

[92] Palo Alto Networks AutoFocus. https://www.paloaltonetworks.com/cortex/threat-intelligence.

[93] Palo Alto Networks Next-Generation Firewall. https://www.paloaltonetworks.com/network-security/next-generation-firewall.

[94] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of internet background radiation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004.

[95] Jong Chun Park and Jedidiah R Crandall. Empirical study of a national-scale distributed intrusion detection system: Backbone-level filtering of html responses in china. In *2010 IEEE 30th International Conference on Distributed Computing Systems*, pages 315–326. IEEE, 2010.

[96] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

[97] Paul Pearce, Roya Ensafi, Frank Li, Nick Feamster, and Vern Paxson. Augur: Internet-wide detection of connectivity disruptions. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 427–443. IEEE, 2017.

[98] Andreas Pitsillidis, Chris Kanich, Geoffrey M. Voelker, Kirill Levchenko, and Stefan Savage. Taster's Choice: A Comparative Analysis of Spam Feeds. In *Proceedings of the ACM Internet Measurement Conference*, pages 427–440, Boston, MA, November 2012.

[99] Jon Postel. RFC0791: Internet Protocol, 1981.

[100] Anirudh Ramachandran, Nick Feamster, and David Dagon. Revealing botnet membership using dnsbl counter-intelligence. *SRUTI*, 6, 2006.

[101] Anirudh Ramachandran, Nick Feamster, and Santosh Vempala. Filtering spam with behavioral blacklisting. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, 2007.

[102] Recorded Future Security Intelligence. https://www.recordedfuture.com/.

[103] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov. Learning and classification of malware behavior. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 108–125. Springer, 2008.

[104] RIPE Atlas - RIPE Network Coordination Centre. https://atlas.ripe.net/.

[105] Martin Roesch. Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238, 1999.

[106] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105. Madison, Wisconsin, 1998.

[107] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D Strowes, and Narseo Vallina-Rodriguez. A long way to the top: Significance, structure, and stability of internet top lists. In *Proceedings of the Internet Measurement Conference*. ACM, 2018.

[108] Karl-Michael Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 307–314. Association for Computational Linguistics, 2003.

[109] Dave Shackleford. Cyber threat intelligence uses, successes and failures: The sans 2017 cti survey. *SANS, Tech. Rep.*, 2017.

[110] Shadowserver. https://www.shadowserver.org/.

[111] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Faith Cranor, Jason Hong, and Chengshan Zhang. An empirical analysis of phishing blacklists. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, 2009.

[112] Rachee Singh, Rishab Nithyanand, Sadia Afroz, Paul Pearce, Michael Carl Tschantz, Phillipa Gill, and Vern Paxson. Characterizing the nature and dynamics of tor exit blocking. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 325–341, 2017.

[113] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of grey: On the effectiveness of reputation-based "blacklists". In *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE.

[114] Snort - Network Intrusion Detection and Prevention System. https://www.snort.org/.

[115] Splunk> SIEM. https://www.splunk.com/.

[116] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[117] Structured Threat Information eXpression. https://stixproject.github.io/.

[118] Sumo Logic Continuous Intelligence Platform. https://www.sumologic.com/.

[119] Symantec Internet Security Threat Report, 2019. https://www-west.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf.

[120] UCSD network telescope. https://www.caida.org/projects/network_telescope/.

[121] The spam and open relay blocking system. http://www.sorbs.net/.

[122] The Spamhaus block list. https://www.spamhaus.org/sbl/.

[123] The Spamhaus Don't Route Or Peer Lists. https://www.spamhaus.org/drop/.

[124] Kurt Thomas, Rony Amira, Adi Ben-Yoash, Ori Folger, Amir Hardon, Ari Berger, Elie Bursztein, and Michael Bailey. The abuse sharing economy: Understanding the limits of threat exchanges. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016.

[125] Threat Intelligence Market Analysis By Solution, By Services, By Deployment, By Application And Segment Forecast, 2018 - 2025. https://www.grandviewresearch.com/industry-analysis/threat-intelligence-market.

[126] Threat intelligence market size by component, by format type, by deployment type, by application, industry analysis report, regional outlook, growth potential, competitive market share and forecast, 2019 – 2025. https://www.gminsights.com/industry-analysis/threat-intelligence-market.

[127] Wiem Tounsi and Helmi Rais. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & security*, 72:212–233, 2018.

[128] University of Oregon Route Views Project. http://www.routeviews.org/routeviews/.

[129] 2020 Best National University Rankings. https://www.usnews.com/best-colleges/rankings/national-universities, Jan 2020.

[130] Giovanni Vigna, Fredrik Valeur, and Richard A Kemmerer. Designing and implementing a family of intrusion detection systems. In *Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 88–97, 2003.

[131] VirusTotal. https://www.virustotal.com/#/home/upload.

[132] Yi-Min Wang, Roussi Roussev, Chad Verbowski, Aaron Johnson, Ming-Wei Wu, Yennun Huang, and Sy-Yen Kuo. Gatekeeper: Monitoring auto-start extensibility points (aseps) for spyware management. In *LISA*, volume 4, pages 33–46, 2004.

[133] Brad Wardman and Gary Warner. Automating phishing website identification through deep md5 matching. In *2008 eCrime Researchers Summit*, pages 1–7. IEEE, 2008.

[134] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE symposium on security and privacy (Cat. No. 99CB36344)*, pages 133–145. IEEE, 1999.

[135] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. 2010.

[136] Min Wu, Robert C Miller, and Simson L Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610, 2006.

[137] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. Panorama: capturing system-wide information flow for malware detection and analysis. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 116–127, 2007.

[138] Zeek Network Security Monitoring. https://zeek.org/.

[139] Le Zhang and Tian-shun Yao. Filtering junk mail with a maximum entropy model. In *Proceeding of 20th international conference on computer processing of oriental languages (ICCPOL03)*, pages 446–453, 2003.

[140] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648, 2007.

[141] Jonathan Zittrain and Benjamin Edelman. Internet filtering in china. *IEEE Internet Computing*, 7(2):70–77, 2003.