AFRL-RI-RS-TR-2022-096

TOWARDS OUTCOME-BASED CYBERSECURITY RISK MANAGEMENT

UNIVERSITY OF TULSA

JULY 2022

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2022-096 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
WILLIAM E. KAHLER
Work Unit Manager

/ S /
JAMES S. PERRETTA
Chief, Information Warfare Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
JULY 2022	FINAL TECHNICAL REPORT	FEBRUARY 2019	FEBRUARY 2022

4. TITLE AND SUBTITLE

TOWARDS OUTCOME-BASED CYBERSECURITY RISK MANAGEMENT

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
	FA8750-19-1-0152	
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
		R2PZ

6. AUTHOR(S)

Tyler Moore

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
The University of Tulsa 800 S Tucker Dr Tulsa OK 74104	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Research Laboratory/RIGA 525 Brooks Road Rome NY 13441-4505	AFRL/RI	AFRL-RI-RS-TR-2022-096

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This final technical report describes the result of a research project investigating economic issues involving the sharing of cybersecurity research data. It describes an effort to investigate data use and production in cybersecurity research publications from 2012-2016. Evidence is presented that researchers regularly use public data as input to research, but only rarely make created data publicly available. Additionally, it is shown that publications that do create datasets and make them publicly available are cited more often than those that do not. Additionally, utilization of the DHS IMPACT platform is investigated. Attributes of datasets are identified that are associated with greater demand from research consumers. Additionally, the value of sharing taken place on the platform is estimated to be approximately $663 million.

15. SUBJECT TERMS

Cybersecurity; incentives; risk management; security economics; network measurement; science of cybersecurity.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	22

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
WILLIAM E. KAHLER	N/A

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES

LIST OF TABLES

1.0 SUMMARY

Cyber risk management by firms is typically driven by evaluating inputs. CIOs and other decision makers use frameworks that compare the set of security controls deployed by their organization to a predefined normative framework (e.g., NIST Cyber Security Framework). Very little cyber risk management is outcome-based -- that is, based on the relationship between security investment and the resulting risk of a breach. Consequently, firms cannot answer basic questions, such as how much security is gained from investing in certain controls? Or, what controls are actually effective in reducing risk?

Several new risk rating services claim to be more outcome-based (e.g., offerings from SecurityScorecard, QuadMetrics, and BitSight). The core idea is to estimate a firm's security level from signals on mismanagement (e.g., open SMTP mail relays) and low-level malicious activities (e.g., spambots) inside the network. The estimated security level is claimed to be predictive of serious breaches. While promising, independent research has yet to confirm any causal link between risk metrics based on external measurements and security outcomes. We also know very little about the validity of these metrics compared to internal measurements of the actual security level. Furthermore, we know next to nothing about how controls influence security levels (as captured by the risk metrics) and security outcomes (as observed in terms of breaches).

This project took several steps towards outcome-based cyber risk management to begin answering questions like those outlined above. In particular, the project made three primary contributions. First, the project constructed a framework that systematized network measurements in the research literature in support of asset discovery. Second, the project conducted rigorous external measurements of security levels. The work focused on accurately measuring the update patterns of public-facing servers running OpenSSH. Third, the project partnered with a Managed Security Services Provider (MSSP) to evaluate the effectiveness of internal rulesets used to trigger network intrusion detection alerts.

2.0 INTRODUCTION

The objective of this research project is to contribute new ways to empirically evaluate the relationship between adoption of cybersecurity controls and secure outcomes (e.g., experiencing a significant incident or not). Because this is such a big challenge at the highest level, the research project has focused on different case studies in which data can be readily obtained by the team. First, we describe an effort to systematically review the existing literature on asset discovery in order to identify how differences in asset discovery techniques can impact the resulting data collected. Next, we develop a novel method for externally measuring how outdated, and consequently how vulnerable, software is on Internet-scale. We focus on the case of OpenSSH due to its ubiquity (installed on over 4 million machines) and an ability to accurately determine through public scanning whether the security patches have been applied. Finally, we describe an effort to internally measure security controls through a partnership with a large MSSP. We examine intrusion detection system rulesets to better understand their use and effectiveness on behalf of the MSSP’s clients.

3.0 METHODS, ASSUMPTIONS AND PROCEDURES

3.1 Methodology for systematizing asset identification

In this section, we first define the necessary terminology and processes to delineate the scope of the literature review. Our definitions include what we consider an asset, what it means to *discover* an asset, and what the formal meta-process of discovering an asset looks like.

We consider as **assets**: (i) all *network identifiers*, e.g., addresses, FQDNs, and contents of DNS zones, and (ii) the *network services* reachable via these network identifiers, defined by the protocol they are implementing, and any information they provide upon initial connect in their banners, e.g., implementation names and version numbers. **Asset discovery** means that the existence of an asset associated with a specific organization becomes, for the first time, known to an entity. **Bootstrapping** is the process of obtaining the initial seed of information that the asset discovery techniques require as input to discover assets.

Finally, from our definition of assets, discovery, the initial bootstrapping step, and the premise of asset discovery via external measurement, we arrive at a model for the formal asset discovery process, shown in Figure 1.

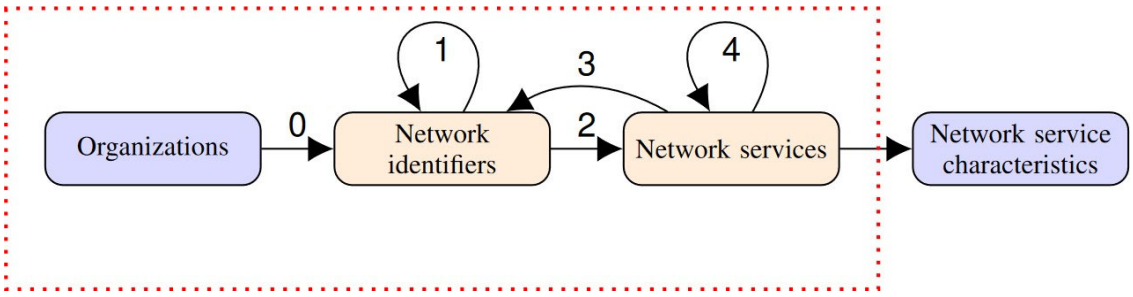

Figure 1. Framework for asset identification techniques.

The **asset discovery process** consists of an initial bootstrapping process, feeding network identifiers into a recursive process to discover more network identifiers (to be fed back into the process) and network services associated with these identifiers. The initial discovery of network identifiers is restricted to the organization of which assets should be discovered (0), which may then yield further associated network identifiers for investigation (1). The discovery of network services associated with network identifiers (2) is straightforward, i.e., one just checks for open ports on the network address (if the identifier is an address) or resolves the identifier to a network address and then checks for open ports. In addition to basic information on the network service, the banners of the detected open ports may then reveal further network identifiers (3) or further network services (4).

We next search the scientific literature for methods and techniques informing or enabling the asset discovery process. We scope our literature search to the major publication venues of Computer Security, Network Measurement, and Network Operations from 2015-2019.

Initially, four researchers independently investigated 940 papers from five years of a leading security conference, ACM CCS, and a leading networking conference, ACM IMC. The researchers were tasked to identify papers that performed asset discovery. They then sought consensus by discussing conflicts, i.e., papers not included or excluded by all researchers. In total, we found 93 papers that utilized asset discovery techniques, as reflected in Table 1. The full methodology, including systematization syntax, is described in a publication (Vermeer et al. 2021).

Table 1: Research papers examined and selected for asset identification, split by venue and year.

	2015		2016		2017		2018		2019		Total	
Security	*Papers*	*Selected*	*Papers*	*Selected*	*Papers*	*Selected*	*Papers*	*Selected*	*Papers*	*Selected*	*Papers*	*Selected*
ACM CCS	128	6	137	2	151	2	134	0	177	2	727	12
IEEE S&P	55	0	55	0	60	1	62	2	84	0	316	3
ISOC NDSS	51	2	60	3	68	0	71	0	89	1	339	6
USENIX Security	67	1	72	2	85	2	100	4	113	1	437	10
RAID	28	0	21	2	21	0	32	1	37	1	139	4
IEEE/IFIP DSN	75	0	65	2	80	1	62	4	54	1	336	8
ACSAC	48	0	48	3	48	1	60	1	60	0	264	5
Networking												
ACM SIGCOMM	40	0	39	0	37	0	40	1	32	1	188	2
ACM IMC	43	2	46	6	42	3	43	3	39	6	213	20
USENIX NSDI	42	0	45	0	46	1	40	0	49	1	222	2
USENIX ATC	47	0	47	1	60	0	76	0	71	0	301	1
IEEE/IFIP NOMS	–	–	222	2	–	–	221	0	–	–	443	2
PAM	27	4	30	1	20	2	20	7	20	0	117	14
TMA	16	0	16	2	29	0	34	1	35	1	130	4
Total	667	15	903	26	747	13	995	24	860	15	**4,172**	**93**

3.2 Methodology for externally measuring security levels

To measure security externally, we can only examine public-facing machines and the information that these machines communicate about themselves. The most relevant and detailed information tends to come in the form of a “banner”, which is the information that is returned by a service running on a particular port. In some cases, the version of a particular software can be identified, which can say something about whether the software is up to date or if it is vulnerable to known exploits.

We utilized Censys to acquire open ports and service banner data across the entire IPv4 address space. Censys keeps historical data, and thus we download several snapshots between 2017 and 2020, which contain banners for FTP, HTTP, HTTPS and SSH.

We also gathered [Censys]. After inspecting these banners, we elected to narrow the scope to just OpenSSH banners running on port 22. The reason we did so is that we could reliably determine in many cases whether servers running OpenSSH had already applied security

patches, even if they were running old software. We downloaded weekly snapshots of the entire IPv4 address space that have SSH banners. These weekly snapshots range from October 2015, which was the earliest we found SSH banners on Censys, through December 2020. We gather software version release dates of several popular Internet-facing software packages from Github and their respective websites and changelogs where available. Security patch release and superseded dates for the OpenSSH software package running on the Ubuntu or Debian Linux distributions are gathered from Launchpad. We acquired OpenSSH patch data dating back to OpenSSH 1.3.8 on Ubuntu Warty (4.10) in 2005.

To study differences in patching among enterprises and cloud providers, we also gathered information on which entities control different portions of IP address space. We gather announced IPv4 address space for several cloud service providers, namely Amazon AWS, Azure Cloud, and Google Cloud. The announced address spaces for these providers are mapped to the IPv4 addresses gathered from Censys. Additionally, we use MaxMind's GeoIP2 dataset and Bureau van Dijk's Orbis resource to identify company ownership for IPv4 CIDR blocks. The full methodology is described in West et al. (2022).

3.3 Methodology for evaluating intrusion detection rulesets

External measurements of security levels often miss out on the security outcome that we were so interested in studying for this project. While in some cases security incidents can be observed externally, they can be much more comprehensively measured when internal measurements are possible. For this portion of the project, we partnered with a Dutch managed security services provider (MSSP). This enables us to evaluate how IDS rulesets are used in the wild for over 100 client firms hired by the MSSP covering more than a decade. We can link the alerts to investigations undertaken by the MSSP, who then confirmed whether an attack had occurred, was blocked from happening, or was a false positive.

The MSSP uses **rulesets** from different origins on the network sensors that they install on their customers' networks. In addition to commercial rulesets purchased from Emerging Threats and VRT, they also employ a proprietary ruleset that is created and maintained by an in-house team of developers. All rulesets are hosted on internal Git repositories. The MSSP has been using its own ruleset since 2008, and the repository has been tracking the changes to that ruleset ever since.

The rules installed on the network sensors produce **alerts** when they detect threats, and all the alerts are logged when they arrive at the SOC. We have access to alert logs that date from mid-2009 to mid-2018. The logs are in CSV format and contain the following information: (i) alert timestamp; (ii) rule ID, revision, and priority; (iii) rule category; (iv) protocol of packet (TCP, UDP, ICMP, IP, numerous application layer protocols); and (v) potential corresponding incident. Between 2017 and 2018, the MSSP was gradually migrating to a different logging platform, causing this logging data to be incomplete throughout that last year.

One or more alerts deemed critical enough by SOC analysts to merit closer investigation are grouped together into an **incident**. The analysts then investigate all related alerts to determine the cause and severity of this incident and ultimately label it accordingly. These labels are

Undetermined, *False positive*, *Not interesting*, *Interesting*, *Low risk*, *High risk*, and *Successful hack attacks*. We consider all but Undetermined and False positive incidents as true positives. The difference between a *High risk* incident and a *Successful hack attack* is that the former involves activities that will directly lead to network compromise, while the latter is an incident where such network compromise has already occurred without the SOC being able to halt the attempt. Undetermined incidents (0.9%) are excluded, as we cannot establish a reliable label for them.

The incident logs contain the following: (i) incident open, response, and close timestamps; (ii) corresponding customer; (iii) category label; (iv) whether the incident was escalated to the customer. This data also dates from mid-2009 to mid-2018 and is also missing entries from 2018 due to the platform migration process. The full methodology is described in Vermeer et al. (2022).

4.0 RESULTS AND DISCUSSION

4.1 Systematization of asset discovery

Of the 93 papers that cover asset discovery, each reported on different edges in the asset discovery process set out in Figure 1. We identified 42 distinct examples of techniques covering edge 1, that is, using network identifiers to discover more network identifiers. For example, domain names were used to identify associated IP addresses or other domain names. We identified 42 distinct examples of techniques covering edge 2, utilizing network identifiers to discover network services running on the device associated with the identifier. This included using domain names to identify mail and name servers, and IP addresses to identify associated services through port scanning. Edges 3 and 4 appeared less often in the literature, only 6 and 8 times, respectively.

This prompts a simple, but interesting, observation that *network service-to-network identifier* (edge 3) and *network service-to-network service* (edge 4) are utilized much less frequently compared to the other two types of asset discovery techniques. Much of the surveyed literature deals with Internet measurement. Given its nature, the initiation of this type of research necessitates basic network identifiers, such as IP addresses and domains.
This logically leads to an overrepresentation of asset discovery techniques that take a network identifier as input.

The techniques used in edge 1 can be roughly grouped into three groups: discovery using 1) passive data sources, 2) functionality of existing technologies and infrastructure, and 3) novel algorithms making use of existing technologies. CAIDA prefix-to-AS and RouteViews data seem to be the standard choice for discovering BGP prefixes and ASs. Both passive and active DNS are widely used in academic research. Edge 2, discovering network services using network identifiers, is the second overrepresented edge. A significant amount of the discovery techniques described in the literature revolve around the network scanner ZMap. Edges 3 and 4 use network services to discover network identifiers and more network services, respectively. Most of the techniques associated with these two edges involve the usage of flaws in configuration or the technology itself.

4.2 Externally measuring software outdatedness in OpenSSH

We first extracted version information from banners. For example, in the banner "`SSH-2.0-OpenSSH_6.7p1 Debian-5+d`", the version is 6.7p1. Our goal was to compute for each IP address how many days behind the version is compared to when the newest version was released. Hence, to track software freshness, we calculate the difference between the snapshot date and when a given software version was *superseded* by a newer version. The days superseded metric more accurately conveys how long the server owner waited to upgrade and is therefore responsible for running outdated software. If a software version is at the latest version at the date of the snapshot, then days superseded is set to 0.

Using the superseded date of the base software version, gives a rather incomplete view of the age of Internet software. Basing software patch levels entirely on the software version information alone may be misleading as it ignores common security practice. Some operating systems will ``backport'' security patches into older versions of a given software without changing the base version number (referred to hereon as the *upstream* version number or upstream patch). In these cases, software may appear to be quite old when looking at the superseded date of the upstream version number even though the security patches are more recent and may fix vulnerabilities which were present in that upstream version.

It is often the case that security patch level information is hidden to all but those with access to the system, which is unfortunate in the case of external measurement via the Internet. Fortunately, we have identified one case where we can reliably observe the presence of backports. In OpenSSH, the security patch version is shown in some banners depending on the configuration of the host operating system, including the popular Ubuntu and Debian Linux distributions.

Consider the example of the following Ubuntu OpenSSH banner string: `SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3`. In this banner, the base version of OpenSSH is 7.6p1, which was released on 2017-10-03. Comparatively, the patch level is 7.6p1-4ubuntu0.3, which was released on 2019-03-04 according to Launchpad. While spoofing these banners is possible, we expect it to be rare since doing so requires editing and compiling the OpenSSH source code. Any edited banners that do not exactly match an Ubuntu backport banner are excluded from the analysis.

Clearly, these patches can be mapped to a much later release date than initially inferred from looking at the upstream OpenSSH version alone. Figure 2 compares the “days superseded” of the Ubuntu security patch level (green line) to the upstream OpenSSH version level (orange line). Now the picture is not only more accurate, it is also a much better outlook from a security perspective. Around 80% of the Ubuntu OpenSSH servers immediately apply patches. If one simply judged software freshness based on the OpenSSH version, 80% of servers would be considered more than three years outdated.

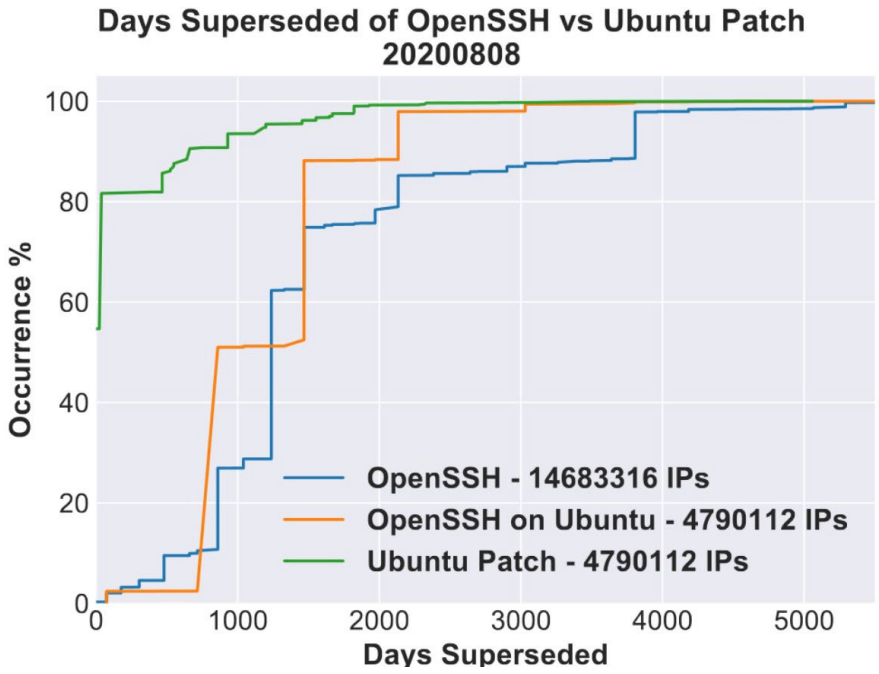

Figure 2. Comparison of days superseded for OpenSSH servers overall, on Ubuntu (upstream version) and Ubuntu (backport version).

From this analysis, we conclude that the picture of server software updates is not as bad as it is often portrayed. We are not out of the woods, though, because 20% of OpenSSH servers are slow to patch. That is a non-trivial number of servers. Moreover, more work needs to be done to connect the application of OpenSSH patches to the presence of software vulnerabilities, which we discuss next.

We look at the distribution of CVEs for OpenSSH over time to test if CVEs influence patch speed. We examined CVEs announced between August 2015 and the end of 2019. For each of these 27 CVEs, we create a mapping of which backported patches are affected by which CVEs. For this mapping, we start by checking which upstream versions are affected by each CVE from the National Vulnerability Database. If the upstream OpenSSH version of a given security patch is not affected by a CVE, then that patch version is not considered to be affected either. For security patches where the upstream version is affected by a given CVE, we inspect the changelog text for that patch, available on Launchpad, to see whether either that patch or a previous patch in its tree claim to fix that CVE. Note that 7 of the 27 CVEs are not fixed in any backports, so the only way to eliminate these vulnerabilities is to manually update the OpenSSH software to a newer upstream version.

We utilize the mapping of CVEs to vulnerable Ubuntu backports in order to compute the number OpenSSH servers that are vulnerable and not vulnerable to each CVE for each Censys snapshot within 2015 and 2019. We then combine these over time to show the total number of IPs which are and are not vulnerable to a given CVE over time. An example of this for CVE-2016-10009 is shown in Figure 3.

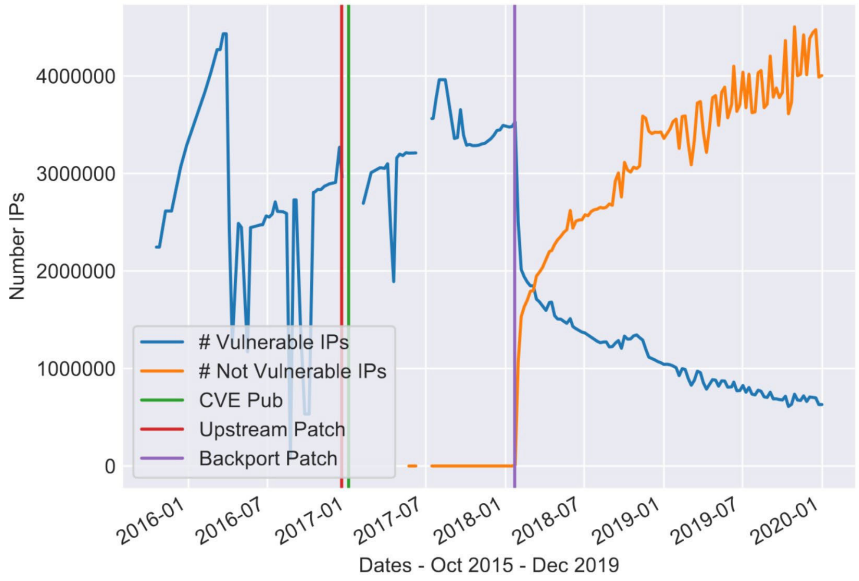

Figure 3. Number of IP addresses that are affected by CVE-2016-10009 over time

For this figure, the blue line represents the number of IP addresses which are vulnerable to CVE-2016-10009 over time, while the orange line is the number of IPs which are not vulnerable. The sum of the orange and blue lines at a given point on the x-axis is equal to the number of servers on that Censys snapshot which are running OpenSSH on Ubuntu with a security patch available on Launchpad. The noise in the table is related to the number of IP addresses Censys scanned at each point in time. The number of Ubuntu servers that Censys scans generally increases over time.

We observe that the near simultaneous publication of the CVE and upstream patch has very little impact on the deployment of vulnerable servers. The steady increase in vulnerable OpenSSH servers continues for slightly more than one year until the Ubuntu backport is published, at which point the patch is rapidly applied to more than one million machines, followed by a steady linear increase in the subsequent months and years. From this one example, at least, it appears that the availability of backports is by far the dominant factor in applying updates to eliminate software vulnerabilities. We analyzed the plots of all 27 CVEs to see if the same trend held true. We see that many IP addresses patch very quickly as soon as a backported patch for a CVE is released, although some never patch. Another consistent finding from inspecting the graphs is that the publication of the CVE and upstream patch is *not* the catalyst for updates. Rather, it is consistently the backport that sparks an uptick in patches to plug vulnerabilities.

We now construct a single, aggregated view of the presence of vulnerabilities in OpenSSH over time. Ultimately, what matters from a security perspective is whether systems have any unpatched vulnerabilities present. We start by calculating the fraction of Ubuntu OpenSSH servers throughout the Internet which are affected by at least one published CVE. We see in Figure 4 that almost all OpenSSH servers are affected by at least one CVE throughout the time of our measurement, saving a small percentage that quickly updated in late 2016. Given that not

all CVEs received a backported patch in Ubuntu, this result is perhaps inevitable, but it is alarming nonetheless.

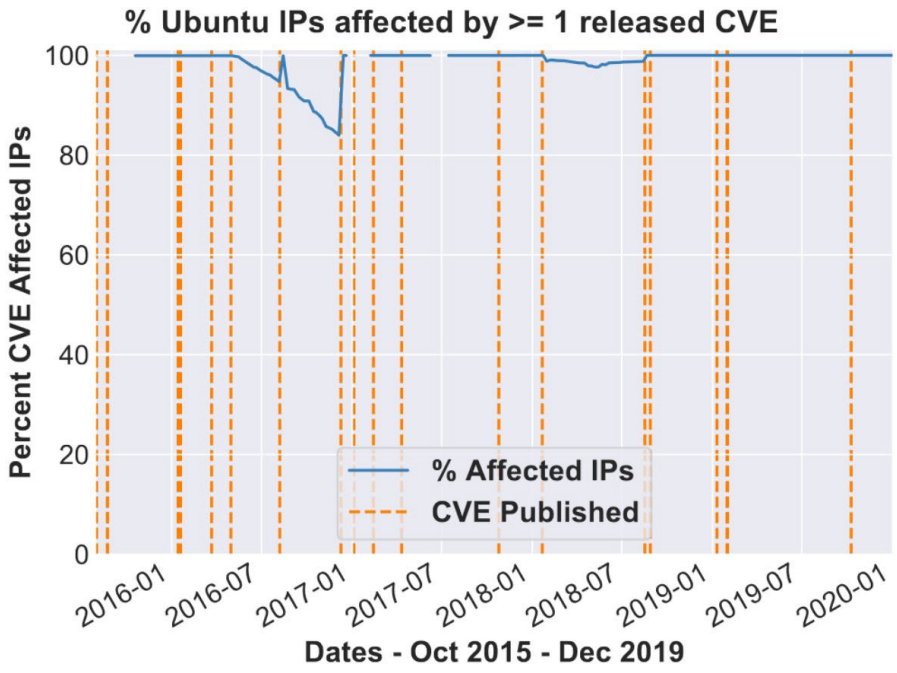

Figure 4. Fraction of Ubuntu OpenSSH servers with vulnerabilities over time.

To account for this, we construct additional measures that focus only on the 20 CVEs which have an associated Ubuntu backport. Figure 5 again plots the percentage of Ubuntu OpenSSH servers that affected by one or more vulnerabilities with an available backport patch.
The percentages here are a bit better, falling to around 60% of hosts vulnerable to at least one CVE before new vulnerabilities are published, rendering all hosts vulnerable until a backport can be issued.

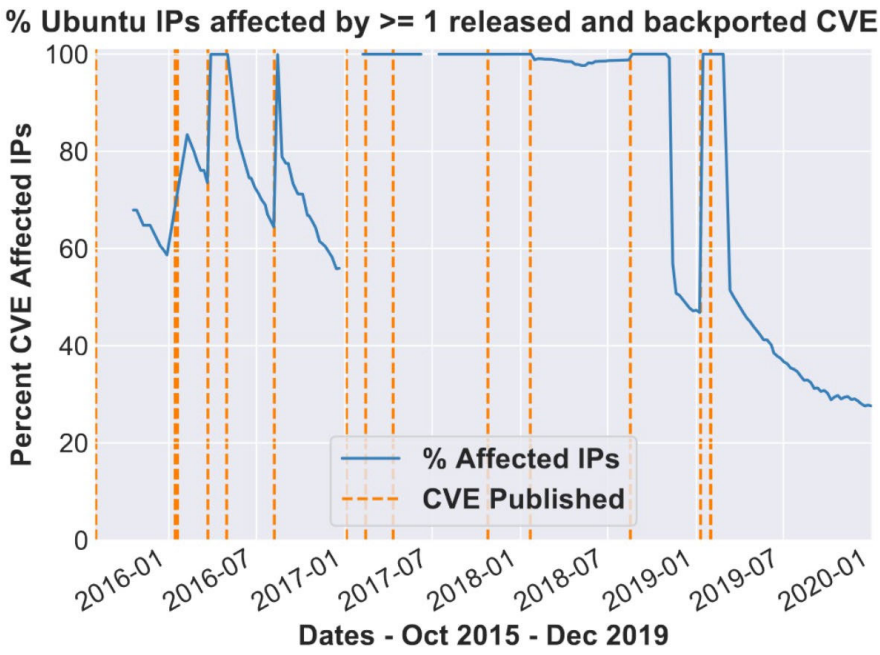

Figure 5. Fraction of Ubuntu OpenSSH servers with vulnerabilities over time (CVEs with available backports only).

Note that for two years in 2016 to 2018 new CVEs were consistently published before backports to the older CVEs were disseminated. The largest reason for this is that several CVEs were released in 2016 which did not receive a patch until 2018, so in the meantime, every LTS version of Ubuntu that was stuck on the upstream OpenSSH version without a security backport was vulnerable. At that point, the only way to not be vulnerable to those CVEs would be to install a fixed upstream version of OpenSSH directly. From the above plots, we can see that no matter how quickly one applies security backports, there is still a chance that the server is vulnerable to at least one CVE.

4.3 Internal measurements of IDS ruleset effectiveness

The MSSP has a particular manner in which alerts arriving at the SOC are processed. Under normal circumstances, all alerts that arrive are processed by the analysts working in the SOC. Deviation from these normal circumstances occurs in cases of false positive floods, for instance. In such cases, the alerts from the responsible rules may be manually suppressed to prevent overburdening the analysts and taking out the SOC back-end systems. In case that one or multiple alerts are suspicious and merit further investigation, these suspicious alerts are grouped together into an incident. Every incident is individually investigated. The analysts determine whether the incident is a false positive, and if so, it is labeled as such. In case of a true positive, the analysts assess the severity of the incident and label the incident.

We first analyze the raw alert data. It simply contains every alert triggered by any active rule present on the probes. Naturally, not every rule will trigger as often as the rest, as some malicious activities are more common than others. We expect this distribution to follow a power-law distribution. Indeed, this is what we find. 672 rules are responsible for 80% of all alerts.

Additionally, out of all the proprietary and commercial rules that we have records of, over 110,000 rules—85%—did not trigger a single alert.

To further investigate the nature of these highly productive rules, we randomly sample and manually examine 50 of the 672 rules. This examination allows us to identify characteristics of the rules that makes them trigger the number of alerts that they do, as oftentimes rules or rule descriptions are unclear, ambiguous, or mislabeled. Of the sampled rules, 52% consists of reconnaissance activity detection and detection of known vulnerability exploitation attempts. This explains the large number of alerts, since these activities are carried out by many a malicious actor on a daily basis across the entire IPv4 space. The remaining 48% not in the two aforementioned categories consists of activities that are not as common an occurrence. Examples include internal network policy violations, DNS requests for malicious domains, or usage of vulnerable software. Therefore, a high level of alerts maintained over a longer period of time is not realistic explanation for the productivity of these types of rules. Indeed, what we find is that 28% of the total sample population are, in essence, quiet rules until a single event causes the rule to trigger up to thousands of times in a single day

Of the aforementioned 672 rules responsible for 80% of alerts, 164 (26%) are proprietary rules, even though proprietary rules make up just 3% of all rules employed by the MSSP that we have a record of. Additionally, of the roughly 20,000 distinct rules responsible for all alerts, around 1,000 rules are from the MSSP's proprietary ruleset. Given its smaller size, the proprietary ruleset performs better than commercial rulesets.

Most of the alerts produced by the rulesets are not deemed relevant or severe enough by the SOC. Of the millions of alerts that the SOC has processed over the years, only a relative handful are investigated more thoroughly: 735 thousand (or 1.2%), which are produced by 6,720 different rules. This subset of alerts add up to 150 thousand incidents. The precise numbers are specified in Table 2.

Table 2. Number of alerts by incident type, with distinct rules triggering alerts.

	#	Associated alerts	Distinct associated rules
Alerts	-	62,321,663	19,744
Incidents	150,437	735,262	6,720
True positive incidents	69,471	674,177	4,731
Risky incidents	13,589	157,388	2,618
Successful hack attacks	106	734	80

Of the 6,720 rules that are present in the incidents, 4,806 (71%) of them are present in 80% of the rules; a very uniform distribution that is in stark contrast to the 672 that produce 80% of all alerts. Interesting here is that of these 672 rules, only 89 are investigated by the SOC, indicating that high alert-producing rules do not necessarily provide usable security information.

Figure 6 illustrates that the number of alerts and (risky) incidents have, not only increased over time, but are also highly correlated. All three curves in the figure are also very much correlated with the increase in the MSSP's customer base. The correlations suggests that organizations remain exposed to external threats to a constant degree as time goes on.

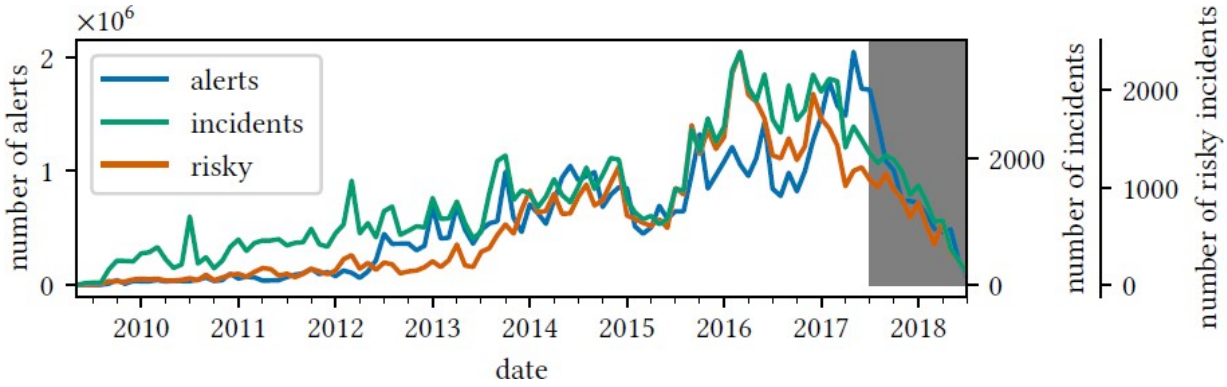

Figure 6. Number of alerts, incidents and risky incidents handled by the MSSP's SOC.

5.0 CONCLUSIONS

This research project has significantly advanced knowledge in terms of bringing empirical measurements to security levels and outcomes from both an internal and external perspective. The project's goal, to work towards providing an empirical basis for evaluating the relationship between cybersecurity controls and secure outcomes, is both important and ambitious. While the project's original plan of funded work encompassed three years, due to a restructuring at DHS, the project duration was shortened to two years. Nonetheless, we were able to make significant contributions that will provide the basis for other researchers continuing to make progress on these important questions.

First, we systematized the research involving asset identification, which forms the basis of all external security measurement investigations. Asset discovery is typically not the main focus of network measurement research. As a result, it often does get the attention it deserves, since enumerating a population of assets involves many trade-offs. It would be far better to explicitly consider the choices and leverage novel techniques proposed by others. However, in practice, an understandable focus on primary measurement tasks, and the lack of a consistent framework to "plug together" asset discovery techniques, often result in an incomplete or ad hoc asset discovery process.

To address these issues, we present a framework for asset discovery. Our framework proposes a syntax to make explicit the steps in the asset-discovery process. This in turn provides a natural way to identify gaps in study design, thereby creating opportunities to build on earlier efforts by broadening the set of assets discovered. Furthermore, our systematization of recent advances in active asset discovery can help researchers select relevant techniques. Finally, we apply our framework to various use cases, which illustrates how techniques can be combined and how to identify where gaps remain.

Second, we contributed a new methodological approach to estimate software outdatedness, which is one of the principal ways to externally evaluate the security posture of organizations. Despite its importance for cybersecurity, measuring the extent to which software is up-to-date at Internet scale has not often been attempted. One reason why is that it is often hard to construct an accurate picture with external measurements. Our project demonstrated that simple approaches to measuring outdatedness based on version information appearing in publicly observable banners often fall short. Instead, we have shown that by focusing on the special cases where we can observe the presence of backports, we can construct a more accurate global measurement for the case of the 4 million-plus servers running OpenSSH on Ubuntu Linux.

We find that these backports do in fact trigger the application of security patches for a significant fraction of the population, much more than vulnerability announcements or updates directly from the software developer. We also observe that when backports are not created, these vulnerabilities tend to remain unfixed for most of the population. Moreover, the frequency of introducing new vulnerabilities has ensured that most servers remain vulnerable most of the time. While we have also presented evidence that cloud providers do a better job, it is not enough to keep hosts running on those platforms from being consistently laden with unpatched vulnerabilities.

Third, we have partnered with an MSSP to comprehensively examine the effectiveness of internal security efforts among the 100+ large enterprise clients of the provider. We focused on the rulesets utilized in intrusion detection, both commercially available and custom-developed for clients in response to perceived threats.

After analyzing four different network IDS rulesets containing around 130 thousand rules, we find that the vast majority of rules fail to produce a single alert, i.e., 80% percent of all alerts were triggered by a mere 0.5% of all rules. However, this does not pose a problem for the SOC analysts as rule developers keep adding new rules and barely modifying the existing ones. In fact, only around 23% of all rules are updated in terms of detection capability, with primarily two objectives: (i) adapt to changes in the threat landscape; and (ii) reducing the number of alerts and false positives by making rules more specific. Hence the possibility of using large rulesets without overwhelming SOC analysts. Just 1.2% of all alerts were deemed important enough to be investigated by the SOC, and only 0.3% of all alerts carried significant risk to the organization.

We also identified a set of common rule management practices that include: (i) using multiple rulesets simultaneously due to the lack of exhaustive coverage of the threat landscape by any single ruleset; (ii) creating proprietary rules to cover client-specific threats and updating these with higher frequency than commercial rulesets; and, (iii) reducing false positive incidents by updating the rules that triggered the corresponding alerts.

6.0 REFERENCES

Mathew Vermeer, Michel van Eeten, Carlos Gañán (2022). Ruling the Rules: Quantifying the Evolution of Rulesets, Alerts and Incidents in Network Intrusion Detection. In *Proceedings of the 17th ACM ASIA Conference on Computer and Communications Security (ACM ASIACCS 2022), pp. 12.*

Matthew Vermeer, Jonathan West, Alejandro Cuevas, Shuonan Niu, Nicolas Christin, Michel van Eeten, Tobias Fiebig, Carlos Gañán, and Tyler Moore (2021). SoK: A framework for asset discovery: Systematizing advances in network measurements for protecting organizations. In *Proceedings of the 6th IEEE European Symposium on Security and Privacy (Euro S&P'21)*, Vienna, Austria (online), September 2021.

Jonathan Codi West and Tyler Moore (2022). Longitudinal Study of Internet-Facing OpenSSH Update Patterns. In *Proceedings of the 23rd International Conference on Passive and Active Measurement (PAM), Proceedings*. Springer-Verlag, Berlin, Heidelberg, 675–689.

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

ACM	Association for Computing Machinery
CAIDA	Center for Applied Internet Data Analysis (CAIDA)
BGP	Border Gateway Protocol
CCS	Computer and Communications Security
CIO	Chief Information Officer
CIDR	Classless Inter-Domain Routing
CSV	Comma Separated Variable
CVE	Common Vulnerability Enumeration
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IMC	Internet Measurement Conference
IPv4	Internet Protocol version 4
MSSP	Managed Security Services Provider (MSSP)
NIST	National Institute of Standards and Technology
SOC	Security Operations Center
SSH	Secure Shell
SMTP	Simple Mail Transport Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol