The CVE Wayback Machine: Measuring Coordinated Disclosure from Exploits against Two Years of Zero-Days

Eric Pauley
University of Wisconsin–Madison
Madison, WI, USA
epauley@cs.wisc.edu

Paul Barford
University of Wisconsin–Madison
Madison, WI, USA
pb@cs.wisc.edu

Patrick McDaniel
University of Wisconsin–Madison
Madison, WI, USA
mcdaniel@cs.wisc.edu

ABSTRACT

Software security depends on coordinated vulnerability disclosure (CVD) from researchers, a process that the community has continually sought to measure and improve. Yet, CVD practices are only as effective as the data that informs them. In this paper, we use DScope, a cloud-based interactive Internet telescope, to build statistical models of vulnerability lifecycles, bridging the data gap in over 20 years of CVD research. By analyzing application-layer Internet scanning traffic over two years, we identify real-world exploitation timelines for 63 threats. We bring this data together with six additional datasets to build a complete birth-to-death model of these vulnerabilities, the most complete analysis of vulnerability lifecycles to date. Our analysis reaches three key recommendations: (1) CVD across diverse vendors shows lower effectiveness than previously thought, (2) intrusion detection systems are underutilized to provide protection for critical vulnerabilities, and (3) existing data sources of CVD can be augmented by novel approaches to Internet measurement. In this way, our vantage point offers new opportunities to improve the CVD process, achieving a safer software ecosystem in practice.

CCS CONCEPTS

• **Security and privacy → Vulnerability management**; • **Networks → Network measurement**.

KEYWORDS

Coordinated Vulnerability Disclosure; Internet Telescopes; Honeypots; Known Exploited Vulnerabilities; Intrusion Detection Systems

ACM Reference Format:
Eric Pauley, Paul Barford, and Patrick McDaniel. 2023. The CVE Wayback Machine: Measuring Coordinated Disclosure from Exploits against Two Years of Zero-Days. In *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23), October 24–26, 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3618257.3624810

1 INTRODUCTION

When bugs in software lead to security vulnerabilities, the way in which these vulnerabilities are handled has a marked impact on risk. As a result, coordinated vulnerability disclosure (CVD) has become essential in ensuring discovered security issues cause limited harm to deployed systems. Yet, CVD practices have limited effectiveness unless they are motivated by a valid and representative understanding of how software vulnerabilities are actually exploited. In response to this concern, members of the community have proposed models [18, 3, 5, 15] for understanding and evaluating the vulnerability lifecycle. Yet, without representative measurement of exploited vulnerabilities it is difficult to reason about the efficacy of CVD and the realism of the model itself.

Formal models are necessary, but not sufficient, for understanding practical CVD. According to one such formal analysis from Carnegie Mellon's CERT Division, when empirically studying CVD "the primary limitation is available observations." [19]. Armed with this formal model, we are therefore faced with a measurement challenge: *how can we collect large-scale, representative data on real-world vulnerability exploitation?* While existing techniques are promising, such as telemetry from antivirus vendors [28] or conventional Internet telescopes [2], each faces limits in terms representativity (e.g., in vantage point or vulnerability type) or access to data (e.g., restricted commercial datasets). Without representative measurement, the software security community lacks a means to evaluate CVD and inform improvements to best practices.

In this work, we use DScope [36], a distributed application-layer Internet telescope we've deployed to Amazon Web Services, to provide a statistical characterization of the lifecycle of high-impact vulnerabilities targeting networked systems. We collect 3 TB of exploit and scanning traffic targeted at 5 M cloud IP addresses from March 2021 to March 2023. Using network intrusion detection system (NIDS) signatures [41], we filter this data to identify *exploit events* that target newly-published vulnerabilities across a broad set of software and vendors. These events, recorded across 63 unique Common Vulnerabilities and Exposures (CVEs), offer a dataset of timelines of *real-world exploitation* of vulnerabilities ranging from end-of-life router firmware to the notorious Log4Shell vulnerability reported in December 2021. We cross-reference six existing data sources to construct complete lifecycles of vulnerabilities from discovery to remediation. Vulnerabilities are diverse across vendors, software weaknesses, and disclosure scenarios, and offer a unique viewpoint through which researchers and practitioners can assess real risk.

Based on measured vulnerability lifecycles, our analysis proceeds in three stages. First, we evaluate whether existing modeling and evaluation of CVD generalizes to our dataset (Section 5). We apply CERT's recently published model [19] to our collected lifecycles and compare results with those seen on a single-vendor dataset from this work. We observe that the heterogeneity of vulnerabilities and

IMC '23, October 24–26, 2023, Montreal, QC, Canada

ACM ISBN 979-8-4007-0382-9/23/10…$15.00
https://doi.org/10.1145/3618257.3624810

disclosure scenarios in our setting leads to less skillful disclosure in general, though CVD still performs more successfully than one would expect from random chance. These results show the promise of larger-scale collection in realistic CVD evaluation.

While real-world data on vulnerabilities allows us to evaluate CVD through CERT's model, we can also use it to evaluate the efficacy *of the model itself* (Section 6). We compare the skill implied by the model to the actual exposure seen by deployed systems. For instance, while a discrete measure of success proposed by prior work concludes that CVD is effective only 56% of the time, when looking at the host level exploit traffic is prevented 95% of the time by existing IDS-based mitigations. Our comparative results show that an exposure-based model of CVD implies higher efficacy than previously thought.

Finally (Section 7), we validate that our findings achieve a useful and general evaluation of CVD. Towards this, we begin by comparing aggregate results from our study with the lifecycles of individual, highly prevalent CVEs in the dataset (including the high-impact Log4Shell vulnerability that occurred during our collection). Our validation through these case studies confirms that overall CVD trends largely apply to the most pervasive and important vulnerabilities.

We also compare our results with reporting of known exploited vulnerabilities (KEV) from the US Cybersecurity and Infrastructure Security Agency. While CISA's KEV benefits from a wealth of manual reports from industry and researchers, we find that, in 50% of cases, a cloud-based telescope measures exploitation of vulnerabilities *over 30 days before* they are incorporated into the KEV. This suggests that Internet telescopes can provide crucial situational awareness to these exploit surveillance efforts, and improve prioritization of vulnerabilities by industry and government. We anticipate that application-layer data from interactive Internet telescopes will prove valuable when used to automatically inform additions to vulnerability repositories such as KEV.

From our work, we come to several recommendations regarding the coordinated disclosure process: (1) Exploitation of critical vulnerabilities can follow very closely after announcement, so automated unsupervised patching of critical software may be necessary to avoid exploitation[1]. (2) NIDS vendors should be more closely involved in coordinated disclosure of vulnerabilities when they could conceivably generate detection and prevention rules. Because NIDS rules can be incorporated into production systems more quickly and confidently than software updates, pushing these rules concurrent with (or before) vulnerability announcement can improve security outcomes. (3) Interactive telescopes should be used in the vulnerability discovery process, especially in detecting the application of known exploit payloads to novel domains. We conclude with a discussion of how security researchers can improve disclosure, and how security and measurement venues can set expectations towards this.

Coordinated disclosure allows vendors to proactively improve their product's security in response to good faith research. Yet, our work shows that understanding of CVD is incomplete without data-driven measurement of effectiveness. We anticipate that our work will motivate new best practices in the reporting, management, and remediation of emergent threats to deployed systems.

[1]This is especially true when updates are low-risk, such as IDS rules.

2 BACKGROUND

When software bugs lead to security vulnerabilities, the way in which these vulnerabilities are discovered, corrected, and published can have a dramatic impact on risk to deployed systems [42]. As such, developers and security researchers follow principles for coordinated vulnerability disclosure (CVD) to reduce the risk of vulnerabilities being actively exploited. While works have modeled this process, such modeling is inherently incomplete without empirical measurement of real-world performance. We briefly discuss background on coordinated disclosure, existing models, and how current data availability limits empirical understanding of CVD performance.

2.1 Coordinated Vulnerability Disclosure

As security researchers discover vulnerabilities in software, these vulnerabilities must be managed in a way that minimizes harm. When disclosure fails, real-world systems are put at risk: for instance, public discussion towards mitigating the recent Log4Shell vulnerability led to in-the-wild exploitation long before end users received formal notification or fixes were made available [16]. To mitigate these impacts, the community has converged on the Coordinated Vulnerability Disclosure (CVD) process [12]. CVD aims to ensure that researchers and vendors share information and mitigations with end-users in a synchronized way, reducing the effective window of vulnerability to attacks.

In the ideal case, coordinated disclosure can result in the complete deployment of mitigations before public knowledge of a vulnerability. This would ensure that a vulnerability is not a risk in practice. For instance, commercial software vendors can auto-install patches before publishing vulnerabilities, a practice followed by major operating system vendors [40, 13, 29]. However, this is not always possible, such as if installations are controlled by users who would not consent to automated updates. The growth of open-source development also poses challenges: when software is developed in public and teams do not adapt to the sensitive nature of vulnerability disclosure, adversaries can discover and exploit vulnerabilities by monitoring development. These challenges necessitate diligence from practitioners, and characterizing past CVD successes and failures can inform best practices.

2.2 Theory of Evaluating CVD

While many have produced models for the CVD process [3, 15, 5], in this work we use the model proposed by Householder et al. of CERT [18, 19]. The CERT model represents the status of a given vulnerability as the result of six events that take place in some given order:

- *Vendor Awareness* (V). The vendor is made aware of the vulnerability, either through internal study, from disclosure/publication by another party.
- *Fix Ready* (F). A fix (e.g., software update) is available that corrects or mitigates the vulnerability.
- *Public Awareness* (P). There is publicly available information about the existence of the vulnerability.

Table 1: Empirical studies of CVE lifecycles. Models used in each paper are converted to those used in this work [18, 19]. Our study combines datasets to characterize the relationships between more CVE lifecycle events and characterizes attack traffic lifecycles on the largest sample of CVEs to date.

Cite	Attack Traffic	# CVEs	Vantage Point	Dates	Events					
					V	*F*	*P*	*D*	*X*	*A*
[3] Arbaugh et al.	✔	3	Common Vulnerabilities	1996-1999	●	●	●	○	●	●
[15] Frei et al.		27 k	Commodity CVEs	1996-2008	○	●	●	○	●	○
[5] Bilge & Dumitraş	✔	18	Antivirus Signatures	2008-2011	○	○	●	○	●	●
[51] Zhang et al.		9	Cloud OS CVEs	2012	○	○	●	●	○	○
[23] Li & Paxson		3.1 k	Open Source CVEs	2005-2016	○	●	●	○	○	○
[1] Alexopoulos et al.		12 k	Open Source CVEs	2011-2020	○	●	●	○	○	○
[18, 19] Householder et al. (CERT)		2.7 k	Microsoft CVEs	2017-2020	○	●	●	○	○	◐[a]
		73 k	Commodity CVEs	2015-2019	○	○	●	○	●	○
This Work	✔	63	DScope-observed CVEs [36]	2021-2023	◐[b]	●	●	◐[c]	●	●

[a] Only considers earliest attack event. [b] Heuristically determined. [c] Deployments modeled based on automated IDS rule deployment.

- *Fix Deployed* (*D*). The fix is adopted in production.
- *Exploit Public* (*X*). Information on how to exploit the vulnerability is publicly known, such as through the publication of a *proof-of-concept*.
- *Attacks* (*A*). Real-world systems with the vulnerability are attacked using the exploit.

Ideally, these events would occur in the order similar to that defined above; vendors would be made aware and develop defenses, the public would be notified and fully deploy countermeasures (though in some cases it may be possible to deploy countermeasures without public awareness), and exploits would be known after systems are fully patched. CERT's model formalizes this ideal in the form of event ordering desiderata: for any two events with respect to a vulnerability, it is desired that one event predate the other (Table 3). The work models CVE lifecycles as a random Markov process with uniformly distributed transitions. This Markov process establishes a certain baseline probability of each desideratum being satisfied by chance (noted as f_d). If a desideratum is satisfied more often than expected by chance, it is said to be "skillful". The CERT model quantifies this skill as $a_d = \frac{f_d^{obs} - f_d}{1 - f_d}$, where f_d^{obs} is the observed frequency of desideratum satisfaction. This definition implies a skill of 0 when achieving the baseline frequency, a skill of 1 when achieving a desideratum 100% of the time, and linearly interpolated as f_{obs} varies in between. This provides a statistical metric for CVD effectiveness that allows for quantitative comparison across time, vantage points, and types of vulnerabilities.

2.3 Prior Empirical Analyses of CVD

To reach practical insights on improving coordinated disclosure, theoretical frameworks must be supported by real-world data. Mitre maintains arguably the most comprehensive listing of security-critical vulnerabilities through their Common Vulnerabilities and Exposures (CVE) program [14], replacing older data sources [8, 22]. Identifiers in this program are also referenced by NIST's National Vulnerability Database (NVD) [31], which provides additional interpretation on the impact of discovered issues. NVD also contains references to relevant sources on the discovery and disclosure process. CISA additionally maintains a catalog of Known Exploited Vulnerabilities (KEV) [21]. Commercial vendors also maintain data on attacks, though contractual and business intelligence concerns complicate use of this data for open research.

Other data sources focus on systematic reproduction of vulnerabilities [30]. For instance, services such as Packet Storm [34] and Exploit-DB [32] provide proofs-of-concept for reproducing exploits against vulnerabilities. These data sources are largely of interest to software vendors who patch vulnerable software or deploy other mitigations, for instance through the publication of signatures for intrusion detection systems (IDS). Exploit databases can be a valuable source of information for adversaries to discover and deploy exploits against vulnerabilities using tools such as Metasploit [27].

2.3.1 Measuring CVE Lifecycles. Based on available data, a variety of works (Table 1) have characterized the CVE lifecycle (the time-series progression of events that affect the status of the vulnerability). These works use sources of CVE metadata [31, 21, 14], combined with additional observations that allow measuring other events in the lifecycle, to characterize a subset of events. For instance, version history for open source projects can allow introduction and remediation of vulnerabilities to be characterized [23]. Such features can also be used to predict the actual risk posed by the vulnerabilities [9], or how long future vulnerabilities may take to discover [1]. Data sources can be augmented by correspondence [43] and surveys of practitioners [49] to understand how vulnerabilities are introduced and discovered.

Our work is informed by prior characterizations of the CVE lifecycle. One seminal work in this space by Arbaugh et al. [3] examined the lifecycle of three prevalent vulnerabilities through manual analysis of specific vulnerabilities. Bilge and Dumitraş [5] achieved larger-scale analysis by using antivirus software analytics, offering a more complete characterization of vulnerability exploitation.

Householder et al. [19] applied the CERT model to empirical lifecycle analysis on bugs in Microsoft products and commodity CVEs. Within these sets of bugs, a subset of CVE events can be extracted, and the relative orderings of these events can be used to estimate CVD skill. Bugs in Microsoft products are especially compelling for analysis, as Microsoft maintains detailed information on the

Table 2: Data Sources. Each source is used to analyze traffic and establish dates for CVE lifecycle events (noted in parentheses).

Dataset	Usage
DScope [36]	Application-layer exploit traffic (A)
Cisco/Talos [33]	Snort Commercial IDS ruleset
Cisco/Talos [45]	Snort IDS rule availability history (F,D)
Cisco/Talos [48]	Talos vulnerability report history (V)[1]
NVD [31]	CVE publication dates and severities (P)
CISA [21]	Known Exploited Vulnerabilities (A)[2]
Suciu et al. [44]	CVE exploit dates & exploitation (X)

[1] Talos-disclosed CVEs only [2] Comparative analysis (Section 7)

history of vulnerabilities, discovery, and known exploitation [47]. However, results from a single vendor may not generalize to the broader software ecosystem.

While these prior works have made promising contributions towards understanding the CVE lifecycle, each faces limitations due to available data, either in terms of number and representativity of vulnerabilities examined, or in terms of the events in the CVE lifecycle that can be determined for analysis. At the same time, questions remain that have not been answerable using existing datasets:

(1) *How vulnerable are deployed systems to zero-day CVEs in practice?* While current works have measured timelines to exploitation for a limited set of vulnerabilities, it is not clear how this translates to actual exposure for deployed services in a realistic setting.
(2) *How does adversarial exploitation vary over time after public disclosure?* Current studies are limited to coarse public data on vulnerabilities and cannot characterize the behavior of adversaries (and therefore risk) in the days, months, and years following announcement of vulnerabilities.
(3) *How do delays in deployment of fixes affect vulnerable systems?* The community is lacking a joint measurement study of exploitation and remediation. Considering these in the context of each other could yield new insights about how CVD can be improved.

Open Problems. While existing measurement studies have made actionable observations into the disclosure process, a measurement methodology that is broad (e.g., across IP address ranges) and representative (across a range of vulnerabilities from many vendors) can yield these more generalizable insights towards improving best practices in coordinated disclosure. Leveraging such a vantage point to characterize vulnerability disclosure is a key focus of this work.

3 METHODOLOGY

We aim to understand how the disclosure, publication, and remediation of vulnerabilities interact with adversarial behavior. To achieve this, we collect application-layer Internet scanning traffic, determine the subset of traffic that targets newfound CVEs, and incorporate additional data on those CVEs to establish the lifecycle of each. We then analyze aggregate trends across vulnerabilities, evaluating the efficacy of both CVD and formal models of it. Finally, we validate our findings on specific vulnerabilities and existing data sources.

3.1 Application-layer traffic & DScope

Our work leverages large-scale collection of Internet scanning traffic. Historically, data sources such as darknet telescopes [7, 35], honeypots [50], or volunteer collection networks [46] would be used to collect similar data. However, these techniques face limits in interactivity (e.g., darknet telescopes do not receive application-layer traffic) or representativity (e.g., honeypots deployed to single IPs will see limited traffic, or volunteer networks may be limited to low-value targets).

To allow for broad interactive collection of Internet scanning traffic, we previously developed DScope [36], a cloud-based Internet measurement apparatus that is designed to be both representative across traffic phenomena and interactive to collect meaningful application-layer data. DScope works by continually allocating new cloud instances on Amazon Web Services (approximately 300 at any given time). Because of the pseudorandom nature of cloud IPv4 address allocation, the IPs of these servers change constantly, with around 30,000 unique IPv4 addresses receiving data each day (DScope collects IPv4 scanning traffic only). These servers are spread across availability zones and regions globally. Once started, each cloud instance accepts TCP connections on all ports. Instances establish TCP sessions but do not send any application-layer response, emulating an unresponsive application-layer service. Each instance runs for a fixed duration, which previous work [36] suggests is optimally around 10 minutes, before terminating and being replaced by a new random IP address. In this way, DScope collects the equivalent of client banner data (as opposed to server-side banner data collected by crawlers). While we extensively consider the ethical implications of this approach in the original work, we include brief summaries of these in Appendix A.

Our previous work established DScope's representativity by comparing collected traffic with that of Merit's ORION [26], a conventional Internet telescope. We demonstrated that the vast majority of traffic seen by conventional telescopes is randomly distributed throughout the cloud IPv4 space, and that DScope therefore achieves representative collection of this traffic. We further identified that a substantial amount of scanning avoids conventional telescopes or targets cloud instances, and that the interactivity of DScope also elicits follow-on traffic from other IP addresses. We concluded that cloud-based telescopes offer a representative vantage point of scanning traffic on the Internet, though specific settings could receive more traffic due to service targeting.

We analyze `pcap` data collected by DScope over the course of 2 years from March, 2021 to March, 2023. This dataset contains 3 TB of application-layer traffic received by 5 M unique cloud IP addresses. Because cloud IP addresses are reused between tenants [24, 38], many of the measured IPs are also previously associated with high-value targets, improving coverage.

Identifying exploit traffic. From the collected traffic, we use Snort [41] to evaluate Cisco-published intrusion detection signatures. We selected Snort and Cisco's ruleset for several reasons: (1) the ruleset has many signatures (>48 k) across varied applications; (2) signatures have known provenance and publication times [45]; and (3)

rules have diverse disclosure scenarios, with some vulnerabilities shared with Cisco before public announcements and others after (this allows measuring the relative effectiveness of coordinated disclosure), including crawlable data on this process [48].

We filter signatures to those matching CVEs published during the study period. We additionally modify all rules so they are port-insensitive. This is necessary because Cisco-published IDS rules are often constrained in specified ports, such that attacks targeted at non-standard ports would not be detected. For each TCP session, we retain only the earliest-published matching IDS signature (of those discussed above). Signatures are evaluated *post-facto* on the entire dataset, allowing retrospective identification of exploit traffic that occurred before public release of signatures or vulnerability reports.

3.2 CVE Root Cause Analysis

In some cases, IDS rules may be unsound, triggering on traffic that does not actually target the vulnerability. For instance, some examined IDS rules triggered on any access to an API endpoint; some traffic would hit this endpoint attempting to brute force credentials. For IDS signatures that matched traffic before they were published, we manually analyzed received traffic to determine if it represented a targeted exploit. CVEs for which IDS rules had false positives (e.g., due to overly general rules) were removed from consideration. The resulting dataset contains 146 k instances of exploit traffic spanning 63 unique CVEs. A complete listing of CVEs analyzed in this work is available in Appendix E.

Identifying CVE disclosure events. We use the NVD/CVE databases to determine events related to the discovery and mitigation of vulnerabilities. As observed in prior works [23, 44], the dates of publication on NVD do not necessarily align with actual public disclosure or knowledge of vulnerabilities. We cross-reference analysis of CVE availability by Suciu et al. [44] to determine best estimates of public knowledge.

Limitations. The vantage point provided by DScope enables us to analyze application-layer behavior of exploit scanners on the Internet. While this provides an opportunity for new analysis, there are important limitations to consider:

(1) *Types of vulnerabilities.* Because we use DScope-collected TCP banner information (sent by clients on initial connection), vulnerabilities are limited to those remotely exploitable via TCP. While studied exploits have high severity (median 9.8 CVSS score and 92nd percentile expected exploitability [44]), this does incur incomplete coverage of security vulnerabilities generally. We argue that this is an acceptable compromise towards achieving an otherwise representative dataset.
(2) *Target Coverage.* DScope is deployed to AWS IPv4 addresses. While the growth of cloud deployments [6] makes these addresses a high-yield target for exploitation [2], DScope may miss traffic targeted at other types of hosts, such as residential IPs. This may, for instance, preference CVEs on software that is more actively maintained, rather than unmaintained software (e.g., residential router firmware). While we compare with a parallel vantage point in Section 7.2, we note that an ensemble of relevant vantage points will prove most useful in providing representative evaluation.

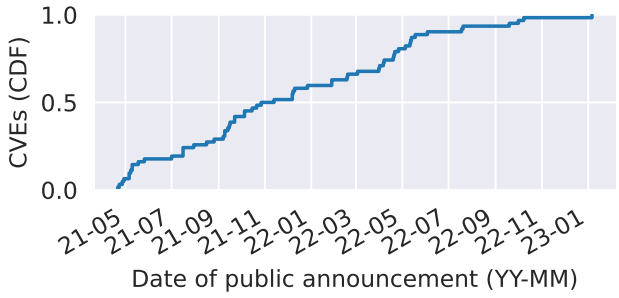


Figure 1: Observed CVEs by public availability [44]. DScope continually observes new emergent threats, a trend we expect to continue in the future.

4 GENERAL TRENDS

Key to our goal of characterizing CVD is a representative vantage point, both in terms overall traffic phenomena and the collection of vulnerabilities studied. We begin by validating our data collection to ensure this representativity.

Our filtered dataset contains traffic targeting 63 unique CVEs. In total, 146 k exploit events are analyzed across these CVEs (application-layer payloads targeting a CVE). Because of the distributed nature of DScope, collected traffic was widespread across receiving IP addresses: 105 k unique telescope IP addresses received analyzed traffic during the study period (out of 5 M total telescope IPs). Note that DScope collects traffic on each unique IP address for only 10 min before allocating a new address, so the fact that most IPs did not receive analyzed traffic is unsurprising. Of the 15 M IPv4 addresses that contacted DScope during the study, traffic targeting new CVEs was sourced from just 3.6 k. This implies that the vast majority of scanning traffic is likely targeting longstanding vulnerabilities or weaknesses not related to specific software bugs (e.g., credential stuffing).

Looking at the distribution of observed CVEs by publication date (Figure 1), DScope receives a steady stream of traffic targeting new vulnerabilities over time. Note that, because this distribution is of CVE publication dates, there is unsurprisingly a drop-off near the end of the study, as CVEs published during this period will have some delay before traffic is seen. We therefore anticipate that the analyses and dataset produced in this paper will be useful for analyzing the evolution of CVD effectiveness over time as more years of data are collected.

Observed CVEs (Appendix E) spanned 40 different software vendors, implying that the trends observed in our analysis should generalize to different organizations. While observed vulnerabilities are scoped by our collection technique to those exploitable over the network, we nonetheless see a diversity of vulnerability type, with 25 CWEs (Common Weakness Enumeration) [25] represented in the data.

Finding 1 - *Observed CVEs skew towards higher-impact vulnerabilities.* Owing to our vantage point (cloud-based interactive Internet

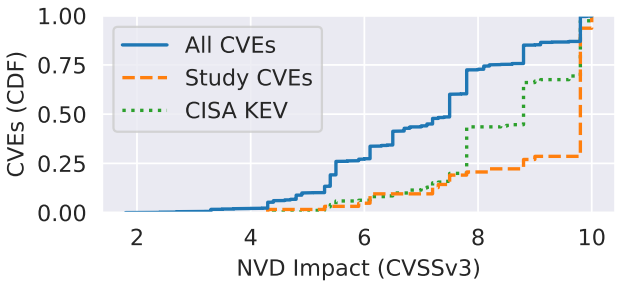


Figure 2: CDF of CVE impact for studied CVEs vs. all CVEs from 2021-2023. KEV CVEs are discussed in Section 7. Studied CVEs skew towards higher impact.

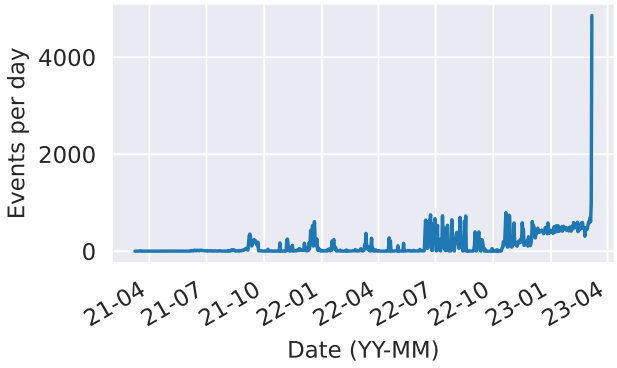


Figure 3: Timeline of CVE exploit events during study. The spike late in the study is caused by a single CVE and does not materially impact the rest of our results.

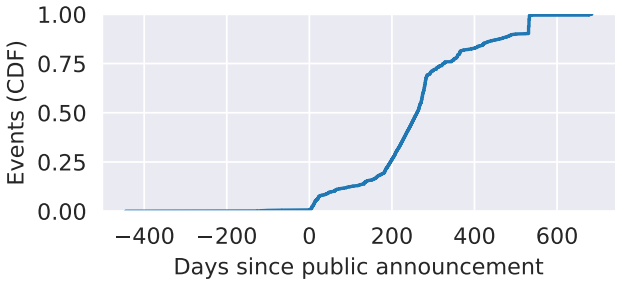


Figure 4: CVE exploit events relative to publication date.

telescope), data is scoped to CVEs that are network-exploitable. This leads to a bias in the data towards high-impact vulnerabilities (Figure 2), though lower-impact vulnerabilities are also represented. We argue that the effect of this on our results is at worst neutral or even positive, as high-impact CVEs have the highest potential for immediate opportunistic exploitation by adversaries, and these vulnerabilities are therefore the most important to study and defend.

Finding 2 - *IDS-based CVE measurement is not limited to IDS vendor disclosures.* One possible concern with using IDS data vendor for CVE detection is that vulnerability lifecycles could be impacted by that vendor's disclosure process. Indeed, Cisco performs independent security research and discloses vulnerabilities in addition to releasing IDS rules. However, the distribution of CVEs in our dataset shows this is not the case: Only 5 of 63 CVEs were originally disclosed by Cisco. We identified 19 CVE assignees across studied CVEs. Assignees ranged from open source repositories and representatives of bug bounty programs (e.g., GitHub, HackerOne), to software vendors (e.g., Apache, Atlassian) and security analysis organizations (e.g., Fortinet, Tenable). We conclude that our dataset is representative across sources of disclosed vulnerabilities.

While the distribution of these CVEs by announcement date is roughly uniform across the study window (Figure 1), we see an increase in raw number of exploit payloads over time (Figure 3). Normalizing traffic relative to CVE publication date (Figure 4) we see an explanation: while there is a spike in traffic targeted at a given CVE immediately after publication, there is still sustained traffic for months or years after fixes have been released. As a result, our dataset contains an increasing rate of matching traffic over time.

Takeaways. From our overall analysis of collected data, we conclude that DScope's vantage point collects a representative sample of cloud-targeted CVE traffic suitable for analyzing coordinated disclosure. Collected data is diverse across targeted system, impact, vendor, and disclosure process, and CVEs are distributed over the course of the study. With the representativity of our collected data established, we next continue by evaluating the efficacy CVD strategies and models.

5 MEASURING CVD EFFECTIVENESS

Based on DScope's dataset, we can measure performance with respect to the CVE lifecycle. For this, we use a framework discussed by Householder and Spring [19] (Section 2.2). Recall that this framework defines 6 events in the CVE lifecycle: vendor awareness (V), fix ready (F), public awareness (P), fix deployed (D), exploit public (X), and attacks (A). Householder and Spring define several desirable orderings of these events (desiderata, Table 3) based on an analysis of the risks posed by each. For instance, it is clearly desirable for Vendor awareness to predate attacks by an adversary. By analyzing our collected data, we can evaluate these desiderata on a representative sample of critical vulnerabilities.

To evaluate desiderata, we proceed by establishing definitive timestamps for various CVE events. This can be done using collected data, third-party datasets, or heuristic combinations thereof:

(1) *(V) Vendor Awareness* is the earliest of public awareness, fix availability, or known disclosure dates (e.g., from Cisco).
(2) *(F) Fix Available* is based on IDS rule availability. Future work could incorporate data on software patches, though this data is not readily available (See Section 8.2).
(3) *(D) Fix Deployed* is based on the assumption of immediate installation of IDS rule updates.[2]
(4) *(P) Public Knowledge* is based on an academic dataset of crawled CVE information sites [44].

[2]Non-commercial users receive IDS rules on a 30-day delay. Given the rapid onset of both attacks and fix deployments after vulnerability publication, these delayed rule updates drastically reduce the effectiveness of IDS.

Table 3: CVE timeline desiderata as presented by Householder and Spring [19] and as restricted by our collection methodology. Cells represent desirability (d), undesirability (u), requirements (r) and impossibility (-) of the row event preceding the column event. For instance, the top-right corner shows $V \prec A$ as desirable. Differences between the two models are caused by inherent orderings in our collection methodology (e.g., $V \prec P$ because our model assumes that public knowledge implies vendor knowledge).

≻	V	F	D	P	X	A
V	-	r	r	d	d	d
F	-	-	r	d	d	d
D	-	-	-	d	d	d
P	u	u	u	-	d	d
X	u	u	u	u	-	d
A	u	u	u	u	u	-

(a) Householder & Spring [19]

≻	V	F	D	P	X	A
V	-	r	r	r	r	d
F	-	-	r	d	d	d
D	-	-	-	d	d	d
P	-	u	u	-	r	d
X	-	u	u	-	-	d
A	u	u	u	u	u	-

(b) This work

Table 4: Satisfaction rate of desiderata on studied CVEs (based on the first time an attack is seen globally). Baseline satisfaction rate is that shown in prior work [19]. The *skill* metric (Section 2.2) provides a metric for CVD efficacy.

Desideratum	Satisfied	Baseline	Skill
$V \prec A$	0.90	0.75	0.62
$F \prec P$	0.13	0.11	0.02
$F \prec X$	0.74	0.33	0.61
$F \prec A$	0.56	0.38	0.29
$D \prec P$	0.13	0.04	0.10
$D \prec X$	0.74	0.17	0.69
$D \prec A$	0.56	0.19	0.46
$P \prec A$	0.90	0.67	0.71
$X \prec A$	0.39	0.50	-0.21

(5) *(X) Exploit public* is determined again based on the same dataset from prior work [44]. This work crawls available web sources to find the earliest public data on CVEs.
(6) *(A) Attack* is based on actual traffic from DScope [36]. Note that this is representative of general exploit scans of public cloud IPv4 addresses, though specific high-profile targets may see exploits earlier.

While our techniques maximize coverage of the CVE lifecycle, there are important limitations. For instance, while DScope traffic does not necessarily establish the first use of an exploit anywhere, the telescope's vantage point on a public cloud provider makes discovered attacks (or lack thereof) far more material to those actually deploying cloud services (an increasingly popular deployment model). Likewise, deployed fixes are often available from other channels (e.g., the original vendor) aside from IDS rule vendors. After establishing CVE timelines for each studied vulnerability, we evaluate desiderata. Based on events dates in the lifecycle, we evaluate desiderata satisfaction for each CVE.

5.1 CVD Skill

Based on relative event orderings, we compute the *skill* associated with each desideratum. Recall (Section 2.2) that the CERT model computes baseline probabilities of desideratum satisfaction. The skill associated with that desideratum is the increase in success compared to that expected under the baseline model. Using this CVD success metric has several desirable properties for our analysis: (1) it allows comparison with initial results in prior work, (2) a quantitative metric for success allows comparison with other scenarios and metrics (Section 6).

Finding 3 - *CVD shows skillful properties even when only considering IDS vendors.* Our results (Table 4), when compared with theoretical values from previous work [19] show that CVD outcomes are skewed towards desirable compared to random sampling. In some cases, results show a large degree of skill: under baseline random behavior we would expect fix deployment to proceed public knowledge only 3.7% of the time [18], but our results show this occurring 13% of the time. Mean skill across all desiderata is 0.37 (recall that 0 implies no skill and 1 implies perfect skill). This implies that CVD as observed through the lens of IDS vendors is currently skillful. Indeed, we see 8 of 9 desiderata satisfied more than expected under the baseline model. Only one set of events ($X \prec A$) shows success rates lower than expected. This ordering represents the desirable ordering that exploit proofs of concept be known publicly before they are employed by adversaries. While this is posited as a desirable ordering in literature, it is weighed as relatively less important than other orderings (e.g., $D \prec A$, fix deployed before attacks).

Finding 4 - *CVD skill is lower than that found by prior work.* While our measurements imply some level of CVD skill, scores are far lower than that observed in prior analysis [19] (finding a skill of 0.969). This is not unexpected, as this previous study focused specifically on $F \prec P$ skill in Microsoft software.

Takeaways. Our analysis demonstrates the effectiveness of CVD broadly across our studied vulnerabilities. At the same time, it establishes a baseline for measuring trends in future vulnerability disclosure. While CVD outperforms expectations as modeled by existing work, performance is relatively poorer when considering the breadth of vendors and vulnerabilities. This shortfall also raises the question of *why* CVD fails, what the impacts are, and how failures could be prevented. To answer these, we next need to improve modeling and measurement of CVD lifecycles.

6 IMPROVING CVD MODELING

While prior modeling examines event orderings, this is not sufficient to understand risk to real systems. In this section, we explore refinements of CERT's model that consider the quantitative timeline of CVD, as well as the relative exposure caused by suboptimal CVD. Here, we hypothesize that the existing model does not sufficiently consider these factors. Further, these adaptations also bring more result explainability, as skill (or lack thereof) can be quantified in terms of actual risk to deployed systems. We posit two new considerations in evaluating the effectiveness of CVD: (1) windows of vulnerability with respect to the CVE lifecycle, and (2) actual exposure of deployed systems to unmitigated exploitation.

6.1 CVD windows of vulnerability

When CVD desiderata are violated (e.g., $A \prec D$), the duration of this violation is material to the actual severity of the risk. Likewise,

even when a desirable ordering is achieved (e.g., $F \prec A$), a longer duration between events offers organizations more time to evaluate fixes and confidently deploy updates. As such, the time-series distribution of events is as important as their relative ordering. For each desideratum, we evaluate the actual time differences between events for each. We graph the CDF of these time distributions across studied CVEs, which then gives insight into when and why desiderata are violated.

Abbreviated results of this analysis are presented in Figure 5 (with additional results in Appendix D). These figures can be interpreted in two parallel ways:

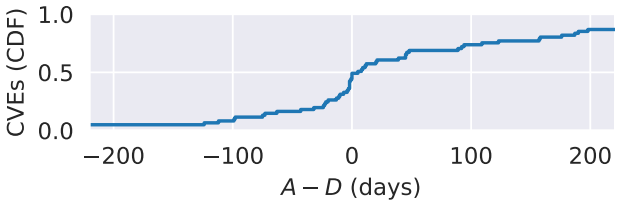
(1) *Relative impact of desideratum (non-)satisfaction.* When a desideratum is met ($diff > 0$), the time between events can be seen as a buffer (e.g., more time between publication and attack traffic allows operators to respond). When a desideratum is not met ($diff < 0$), the negative duration can be seen as a window of vulnerability during which systems are put at risk.
(2) *Hypothetical desiderata scenarios.* The CDF at $diff = 0$ represents the proportion of CVEs where a desideratum was unsatisfied. Shifting this CDF right by some x days shows a hypothetical scenario of performance across CVEs being improved by x days.

6.1.1 Defense Deployment. Arguably the most important component of CVD is rapid deployment of mitigations to deployed systems. By examining the distribution of this deployment D over time (with respect to public knowledge P of the CVE). We begin by analyzing the deployment of fixes with respect to adversarial exploitation (Figure 5a). In comparing this distribution with publication-relative deployment (Figure 5b), we also reach recommendations towards improving CVD.

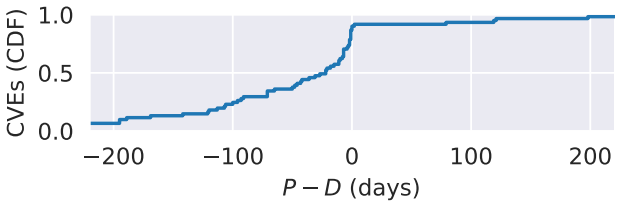
Finding 5 - *Defense deployment often narrowly fails the $D \prec A$ desideratum.* When attacks precede defenses ($A - D < 0$ in Figure 5a), they often do so by a very brief period (only a few days). While the Householder and Spring model classifies this as a failure, it is not as severe as a longstanding vulnerability without deployed mitigations. This brief window also implies that minor improvements to the CVD process could yield outsize protection in practice. To better understand how this brief shortfall occurs and possible improvements, we next examine the relationship between deployment and public knowledge.

Finding 6 - *Defense deployment very closely follows public availability in many cases.* Looking at Figure 5b, we see a large mass of CVEs with IDS-based fixes published very shortly (within 10 days) following public availability. This implies that IDS vendors are not included in the private coordinated disclosure process, yet those vendors are highly adept at creating fixes after a vulnerability is known to them. Only 8 (13%) of the CVEs studied had IDS-based fixes deployed before publication. 5 of these were vulnerabilities discovered by an affiliate of the IDS vendor.

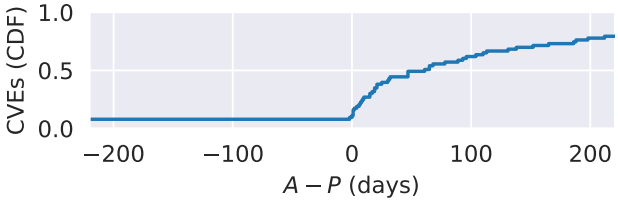
When deployment of fixes by IDS vendors very closely follows publication, we posit that the IDS vendor acted in response to publication, rather than privileged access to advance notice from vendors. To analyze the effect of this, we consider hypothetical CVE timelines where the IDS vendor would be included in disclosure.



(a) $A - D$ **(**$P(D \prec A = 0.56)$**). Attacks often closely (**< 30 **days) precede deployment of fixes (**$A - D < 0$**), but more rarely closely follow (**$A - D > 0$**).**



(b) $P - D$ **(**$P(D \prec P) = 0.13$**). Deployment of fixes rarely precedes public awareness, though it often follows by only brief (**< 10 **days) periods.**



(c) $A - P$ **(**$P(P \prec A = 0.90)$**). Duration from publication to attack follows a rough exponential distribution.**

Figure 5: Time-series representation of desiderata. Each plot is a CDF of the time difference between events.

For CVEs where IDS mitigation occurred within 30 days of public announcement, we modify that date to be equal to publication date (assuming that IDS rules would be published alongside this announcement). This assumption is in-line with actual observed disclosures that included IDS vendors, where rules were released before vulnerability publication.

Finding 7 - *Subtle shifts in the CVD timeline can lead to drastic improvement in performance.* As a result of this experiment, our CVE shows substantial improvement in the $D \prec A$ desideratum (recall that this refers to fix deployment predating attacks; this is arguably the most important desideratum). Desideratum satisfaction increases from 0.54 to 0.65, and associated *skill* value improves by 32%. Our new proposed model and associated findings demonstrate the potential for improvement in the CVD process.

6.1.2 Attack timing. **Finding 8** - *Attack traffic closely follows publication.* Figure 5c shows time differences between publication and attack traffic. The few attacks in the dataset that precede publication do so by long durations, implying that attacks are performed without vendor awareness, while those that follow publication often do so closely. However, unlike prior work [5], which concluded that 93% of vulnerabilities were exploited in the first week

Desideratum	Satisfied	Baseline	Skill
$V \prec A$	≈ 1.00	0.75	0.99
$F \prec P$	0.01	0.11	-0.11
$F \prec X$	0.54	0.33	0.31
$F \prec A$	0.95	0.38	0.92
$D \prec P$	0.01	0.04	-0.02
$D \prec X$	0.54	0.17	0.45
$D \prec A$	0.95	0.19	0.94
$P \prec A$	0.99	0.67	0.98
$X \prec A$	0.95	0.50	0.91

Table 5: Rate of desideratum satisfaction on a per-exploit-event basis.

after publication, we saw a much shallower trend. We attribute this to the different class of vulnerabilities (client-side malware vs. network-based attacks) studied, both because malware tends to target larger-scale deployments such as operating systems, and because a network-based vantage point is limited by its footprint.

6.2 Quantitative System Exposure

While the baseline model assumes that each CVE event occurs at a single point in time, this is often far from true in practice. Users install patches on a delayed timescale [39], and individual exploit events may not mean that all systems are rapidly compromised. In this section, we compute CVD event orderings on a per-event (instance of CVE-targeted traffic to a given IP address) basis, rather than a per-CVE basis. In this way, we account for differences between first-exploitation and exploitation generally.

Finding 9 - *CVE exploitation is concentrated after publication.* Figure 6 shows the number of CVEs targeted in 5-day windows after publication. CVEs are primarily targeted in the days and weeks after publication, though others see sustained targeting over time. This could be influenced by factors such as ease of updating vulnerable systems and overall deployment size (both leading to potential large residual unprotected populations), as well as re-exploitation potential (i.e., whether systems are valuable targets for exploitation after having already been exploited; Mirai is one such example where reinfection targeting is valuable [2]).

Finding 10 - *Considering actual exposure, CVD is highly effective.* Table 5 shows desideratum satisfaction rates on a per-event basis. We see high effectiveness of $D \prec A$ (deployed fixes before attacks) of 95%, compared to only 52% when aggregating across CVEs. This implies that existing models understate the effectiveness of CVD in practice.

6.2.1 Measuring mitigated exposure. We can also measure CVE exposure by analyzing the distribution of mitigated and unmitigated deployments. Here, we take all CVE exploit events, normalize timestamps relative to public knowledge date, and segment traffic by whether the targeted vulnerability had a deployed defense at the time. We present this data in two ways: binned by unique CVE and shown as a histogram (Figure 6), and as a CDF of overall events (Figure 7). These allow reasoning about the diversity and intensity, respectively, of CVE exposure and mitigation.

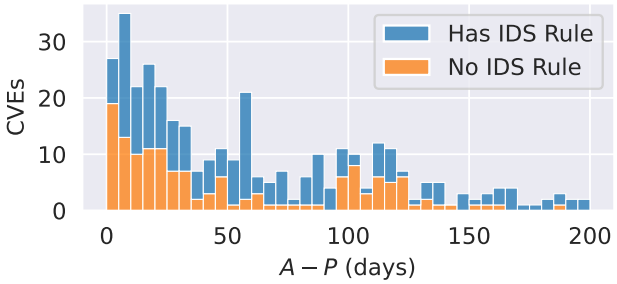


Figure 6: Number of CVEs observed relative to publication time. CVEs are separated based on whether an IDS rule is available during that bin.

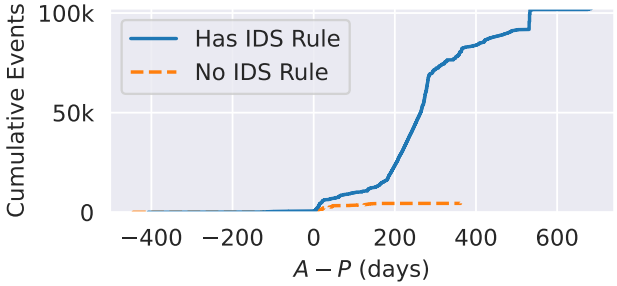


Figure 7: CDF of overall CVE exploit events over time since public disclosure, segmented by whether an IDS signature would have blocked the traffic.

Finding 11 - *Deployed fixes cover the majority of targeted CVEs within 5 days.* Looking again at Figure 6 we see that of the diminishing number of unique CVEs targeted over time, the share of vulnerabilities with mitigations rapidly increases. Beyond the first 5-day bin IDS rules cover the majority of actively exploited vulnerabilities. This effect does not hold in all later bins, suggesting there may be adversarial adaptation towards targeting unmitigated vulnerabilities.

Finding 12 - *Unmitigated exploitation is concentrated after public disclosure.* While Finding 9 found some concentration of CVE exploit events after publication, when segmenting for *unmitigated* CVEs this concentration is more pronounced. Figure 7 shows the cumulative count of overall mitigated and unmitigated CVE exploit events. Notably, 50% of unmitigated exposure occurs in the first 30 days after publication. This quantitative-exposure result is also consistent with that of Finding 7, suggesting that modeling CVEs through windows of vulnerability and quantitative exposure yield comparable and complementary results.

Takeaways. By analyzing CVD through the lenses of windows of vulnerability and actual exposure, we observe parallel confirmation of several significant trends. First, because the of the concentration of unmitigated attacks directly after publication, deployment of fixes is highly salient to CVD skill. Even a short delay in deployment can lead to substantial harm. We discuss implications on the disclosure process in Section 8.

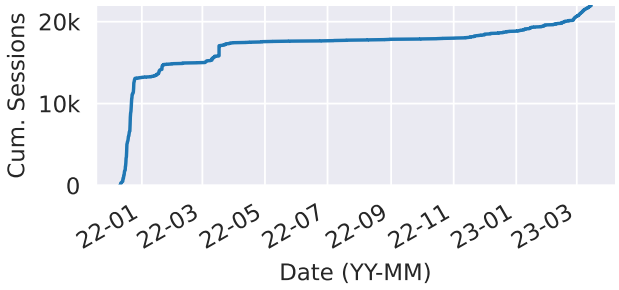


Figure 8: CDF of Log4Shell TCP Sessions over time.

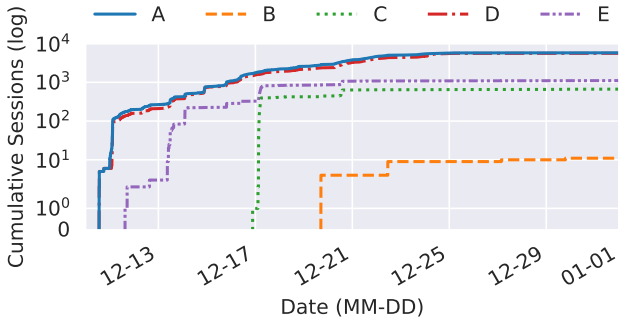


Figure 9: CDF of Log4Shell traffic variants over time during the month of December, 2021. Each series is a group of IDS signatures released at the same time (Table 6). Results show increased attack sophistication and targeting over time.

7 VALIDATION

While our results demonstrate new methods to evaluate CVD, they raise a natural question: *do aggregate analyses reach correct conclusions with respect to individual CVEs?* A key benefit of performing CVE analysis on telescope data is the ability to look retrospectively. We select two high-profile CVEs that occurred during our measurement period for such a retrospective analysis. We evaluate the Log4Shell vulnerability in Apache's Log4J library, and perform a parallel study of vulnerabilities in Atlassian's Confluence in Appendix C. We examine whether trends are consistent with those seen in the larger population. We additionally demonstrate how DScope's application-layer traffic collection enables fine-grained characterization of vulnerabilities. We also compare results from our collected data to an existing corpus of known exploited vulnerabilities. In so doing, we highlight how each methodology leads to complementary results.

7.1 Log4Shell (CVE-2021-44228)

Background. In November 2021, a security research group reported a command injection vulnerability in Apache's Log4J library [16]. This vulnerability, known as Log4Shell, allows user-passed input to invoke escape sequences that can download and execute arbitrary code from the Internet. While there have been reports of small-scale attacks shortly after pull requests appeared on the project's GitHub repository, official public disclosure of the vulnerability led to widespread Internet scanning and exploitation. As noted in prior analysis [16], Log4J's inclusion as a library in a variety of other distributed applications makes the vulnerability especially difficult to correct, as copies of the library have been broadly distributed.

Analysis. Log4Shell's diverse exploitation makes it a strong candidate for validating our results. We examine traffic for the CVE generally, then look at specific vulnerability variants.

Finding 13 - *Log4Shell shows rapid exploitation after public disclosure, with reduced targeting over time.* Figure 8 shows the number of CVE exploit events against all Log4Shell vulnerabilities over time. In many ways this distribution aligns with our broader measurements: high event density occurs early on after exploitation, with lower sustained density over time. Interestingly, we see a resurgence in exploit events nearly a year after initial publication. Exploration of this is beyond the scope of this work, but we hypothesize that the intended targets have shifted from high-profile services (as in the initial phases of Log4Shell [20]) to legacy services and embedded systems that are more broadly distributed and unpatched.

Finding 14 - *Log4Shell exploits showed increasing sophistication in the days following public release.* A somewhat unique quality of the Log4Shell vulnerability is that it can be exploited through a wide variety of channels. Because of Log4J's use as a third-party library, exploit payloads can reach the vulnerable code directly, or through a set of preprocessing steps. Because of this, initial naive attempts to create intrusion detection rules for Log4J vulnerabilities were limited. Adversaries can take advantage of the preprocessing step of programs to create exploit payloads that bypass naive detection while still triggering vulnerabilities when decoded. Figure 9 depicts the distribution of exploit TCP sessions across exploit variants. In the days following release, public knowledge of the vulnerability allowed for rapid development of new exploit variants that necessitated rapid response.

Takeaways. Analysis of Log4Shell conforms to our broader conclusions over studied CVEs. Software vendors should be aware that, when publicly releasing information on novel vulnerabilities, exploitation will advance rapidly following release. Tiered disclosure to privileged participants can improve the probability of rapidly deployable fixes being available to affected parties before active exploitation occurs.

Results also suggest a seemingly fundamental limitation of IDS signatures. While signatures aim to detect the *semantics* of a vulnerability rather than just a specific instance, interactions between vulnerable software and other dependent components can lead to unpredictable phenomena. In Log4Shell, increasing adversarial sophistication thwarts IDS-based defenses when they target the symptom of a vulnerability rather than the underlying issue. This is a complex problem to solve, as broader matching for exploits can lead to false positives.

7.2 Comparison with Existing Attack Data

We next compare results seen by DScope with CISA's Known Exploited Vulnerability (KEV) catalog [21]. This database contains

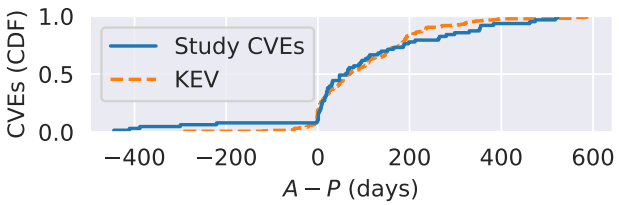


Figure 10: $A - P$ for Known Exploited Vulnerabilities.

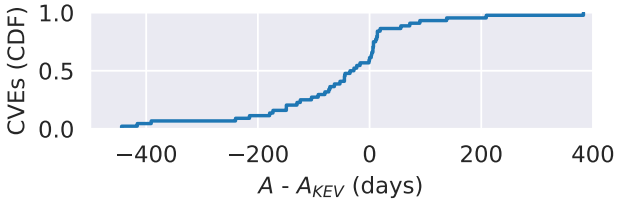


Figure 11: Difference between earliest exploitation as seen in our study vs. documented in CISA KEV. Negative values indicate DScope-observed exploitation before CISA KEV; positive values indicate a delay in DScope-observed exploitation.

vulnerabilities that are known to be actively targeted by attacks. KEV was initially started part-way through the study period in November 2021. While the primary purpose of this is to prioritize and set expectations for remediations by organizations, we can also leverage these results to analyze CVE lifecycle trends. We filter KEVs to those matching CVEs published during the study period to allow for direct comparison. A total of 424 such CVEs are considered.

Finding 15 - *CISA KEV shows similar bias towards high-impact vulnerabilities.* Figure 2 shows distributions of CVEs from KEV, DScope, and the overall population. KEV shows bias towards high-impact vulnerabilities, but not as strong of a bias as that seen by DScope.

Next, we compare the lifecycles of CVEs seen in KEV and our dataset. Because available data on KEV is limited, we focus on the relation between CVE publication and known attacks. A vulnerability is considered published when the CVE is available on NVD, and exploited when its entry was added to KEV. We compare both the overall distribution (Figure 10) and $P \prec A$ desideratum between the two datasets.

Finding 16 - *Telescope-based CVE characterization sees a higher incidence of attacks long before publication.* While DScope saw a lower rate of exploitation before publication ($A \prec P$) than KEV (10% vs. 18%), we observe that DScope sees a higher rate of CVEs being exploited a long duration (i.e., hundreds of days) before publication. We attribute this to two factors: (a) CVEs that are inadvertently exploited by adversaries targeting other software (as in Appendix C) and (b) earlier instances where unsophisticated vendors failed to respond in a timely fashion to ongoing exploitation.

Validating cloud-based characterization of A. We next compare dates of exploitation as seen by our study with those recorded in CISA's KEV. Of the 63 CVEs in our study, 44 (70% were also present in KEV. The other 30% were observed by DScope but not known-exploited in existing data. For CVEs seen in both datasets, we compute the difference between first exploitation times (earliest traffic time in our study vs. date added in KEV) and plot the distribution (Figure 11).

Finding 17 - *DScope finds evidence of exploitation before existing data sources.* Because CISA KEV aggregates actual reports from many parties, we expectedly see a delay from CISA KEV to telescope-observed exploitation in some (41%) cases. Perhaps more notable, however, is the sizable share (26, or 59%) of CVEs that are seen in the cloud setting first. In fact, 50% of CVEs are observed by DScope over 30 days before they are added to KEV. We attribute these both to (a) vulnerabilities exploited before publication ($A \prec P$), (b) administrative delays in updating KEV based on reports, (c) lack of available data on exploitation. We see CISA's manual report and our traffic-driven approach as complementary tools for characterizing known exploited vulnerabilities and recommend increased use of automated traffic analysis in studying CVE trends.

8 DISCUSSION AND FUTURE WORK

As new vantage points (such as DScope) increase the granularity of data collection, the community has the opportunity to leverage these to achieve practical improvement in the security of deployed systems. We foresee improvements in two concrete areas: (1) expectations for reporting vulnerabilities, (2) data collection to enable future evaluation of CVD.

8.1 Improving CVD

A key finding of our work is the importance of remediation around public disclosure. When vendors control publication, this should be carefully coordinated–both with customers and other parties such as IDS vendors–to maximize deployment of fixes relative to public knowledge. In some cases, however, organizations have no choice in the timing of public disclosure. Both malicious actors and good faith security research can also pose risks. Many measurement and security venues establish standards for vulnerability disclosure timelines, including delayed publication, but when vendors are unsophisticated these timelines may be too tight to ensure a successful outcome.

In the context of lack of vendor sophistication [16] and delayed deployment of security updates by system administrators and end users [39], our analysis suggests that vendor-based disclosure on its own may not be sufficient to minimize harm. Our data rather suggest that IDS vendors in particular are well-suited to provide defenses. Signature-based IDS is as beneficial as ever (though now often repackaged in the form of offerings such as web application firewalls). Researchers disclosing vulnerabilities should consider vendor sophistication, and how additional parties could be included in the disclosure process to achieve the best possible outcome.

Our recommendations also apply to the expectations set by academic venues. Although calls-for-papers of major security and measurement venues largely have expectations of responsible disclosure [37], the implementation of this disclosure is only vaguely

described. Venues should incorporate prescriptive expectations on the form and scope of disclosure to ensure that discovered vulnerabilities minimize harm.

8.2 Improving measurement of CVD

Empirical characterization of CVE timelines has been historically limited by available data, especially with respect to the internal disclosure process and deployment of mitigations. While our work makes promising steps in maximizing coverage, these limitations still prevent complete characterization.

Here, we see an opportunity for security researchers to improve not only the impact of their disclosure process on deployed systems, but also to contribute to the community's understanding of the CVD process. Just as implementation artifacts have increasingly become preferred from published papers, researchers should release disclosure artifacts: data (ideally in machine readable form) that summarizes the steps taken during the disclosure process. We see the following data as being most critical to future characterization of CVD:

- (*V*) *Disclosure date(s).* When and to whom initial disclosure was made, e.g., software/hardware vendors, operating systems, IDS rule vendors, governments, etc.
- (*F*) *Fix development.* For each disclosure party, timeline for development of fixes. Data should describe the scope of fixes (e.g., software vendors can directly fix issues, library vendors may be more limited).
- (*D*) *Deployment.* Fine-grained data on fix deployment would aid in assessing risk. In some cases (e.g., hosted software), an exact characterization of deployment can be provided. In others (e.g., operating systems and web browsers), characterization may be possible using vantage points such as web analytics.
- (*A*) *Known exploitation.* Current known attack data (e.g., CISA KEV) only tracks when attacks are reported, which also limits tracking to published CVEs. When attacks are known before publication or retrospectively, adjusted timing and impact should be reported.

We foresee tracking of these disclosure artifacts being managed by existing organizations, such as the CERT Coordination Center (CERT/CC) [11]. While CERT/CC tracks and releases information on the observed status of vulnerabilities, sharing this additional data from the security researcher's perspective would augment this data and make it more useful for analysis. We see security and measurement venues playing a role here by setting expectations. In addition to encouraging authors to maximize the beneficence of their disclosure process, calls-for-papers should encourage standardized submission of CVE metadata to coordinating authorities.

9 CONCLUSION

By leveraging DScope's perspective, combined with time-series modeling of CVE lifecycles, we provide new insights on the disclosure process. We evaluate two new empirical models for evaluating the efficacy of CVD, and as a result offer recommendations for improving the disclosure processes of security researchers and academic venues. We anticipate that our collection methodology will offer continued value over time, and plan to make interactive telescope-based analysis of emergent threats publicly available. We anticipate that ongoing improvements to empirical results in CVD will inform best practices to provide practical protection to deployed systems.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE1255832. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1900873, CNS-2039146, CNS-2106517, CNS-2312709 and CNS-2319367. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Nikolaos Alexopoulos, Manuel Brack, Jan Philipp Wagner, Tim Grube, and Max Mühlhäuser. 2022. How Long Do Vulnerabilities Live in the Code? A Large-Scale Empirical Measurement Study on FOSS Vulnerability Lifetimes. en. In 359–376. ISBN: 978-1-939133-31-1. https://www.usenix.org/conference/usenixsecurity22/presentation/alexopoulos.

[2] Manos Antonakakis et al. 2017. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*, 1093–1110.

[3] William A. Arbaugh, William L. Fithen, and John McHugh. 2000. Windows of vulnerability: A case study analysis. *Computer*, 33, 12, 52–59. Publisher: IEEE.

[4] 2022. Atlassian Confluence Vulnerability CVE-2022-26134 Abused For Cryptocurrency Mining, Other Malware. en-US. (Sept. 2022). https://www.trendmicro.com/en_us/research/22/i/atlassian-confluence-vulnerability-cve-2022-26134-abused-for-cryptocurrency-mining-other-malware.html.

[5] Leyla Bilge and Tudor Dumitraş. 2012. Before we knew it: an empirical study of zero-day attacks in the real world. en. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, Raleigh North Carolina USA, (Oct. 2012), 833–844. ISBN: 978-1-4503-1651-4. DOI: 10.1145/2382196.2382284.

[6] Todd Bishop. 2021. Amazon Web Services posts record $13.5B in *profits* for 2020 in Andy Jassy's AWS swan song. en-US. (Feb. 2021). https://www.geekwire.com/2021/amazon-web-services-posts-record-13-5b-profits-2020-andy-jassys-aws-swan-song/.

[7] Elias Bou-Harb. 2015. *Approaches and Techniques for Fingerprinting and Attributing Probing Activities by Observing Network Telescopes*. en. phd. Concordia University, (June 2015). https://spectrum.library.concordia.ca/id/eprint/980132/.

[8] [n. d.] Bugtraq Mailing List. (). https://seclists.org/bugtraq/.

[9] Benjamin L. Bullough, Anna K. Yanchenko, Christopher L. Smith, and Joseph R. Zipkin. 2017. Predicting Exploitation of Disclosed Software Vulnerabilities Using Open-source Data. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics* (IWSPA '17). Association for Computing Machinery, New York, NY, USA, (Mar. 2017), 45–53. ISBN: 978-1-4503-4909-3. DOI: 10.1145/3041008.3041009.

[10] Andrew Case, Sean Koessel, Steven Adair, and Thomas Lancaster. 2022. Zero-Day Exploitation of Atlassian Confluence | Volexity. en-US. (June 2022). https://www.volexity.com/blog/2022/06/02/zero-day-exploitation-of-atlassian-confluence/.

[11] [n. d.] CERT Coordination Center. (). https://www.kb.cert.org.

[12] [n. d.] Coordinated Vulnerability Disclosure Process | CISA. en. (). https://www.cisa.gov/coordinated-vulnerability-disclosure-process.

[13] Jonathan Corbet. 2022. What constitutes disclosure of a kernel vulnerability? [LWN.net]. (June 2022). https://lwn.net/Articles/896829/.

[14] [n. d.] CVE - CVE. (). https://cve.mitre.org/.

[15] Stefan Frei, Dominik Schatzmann, Bernhard Plattner, and Brian Trammell. 2010. Modeling the security ecosystem-the dynamics of (in) security. *Economics of Information Security and Privacy*, 79–106. Publisher: Springer.

[16] Raphael Hiesgen, Marcin Nawrocki, Thomas C Schmidt, and Matthias Wahlisch. [n. d.] The Race to the Vulnerable: Measuring the Log4j Shell Incident. en, 9.

[17] S. Hills. 2013. Considerations and recommendations concerning internet research and human subjects research regulations, with revisions. *HHS. gov.*

[18] Allen Householder and Jonathan Spring. 2021. A State-Based Model for Multi-Party Coordinated Vulnerability Disclosure (MPCVD).

[19] Allen D. Householder and Jonathan Spring. 2022. Are We Skillful or Just Lucky? Interpreting the Possible Histories of Vulnerability Disclosures. en. *Digital Threats: Research and Practice*, 3, 4, (Dec. 2022), 1–28. DOI: 10.1145/3477431.

[20] Tatum Hunter and Gerrit De Vynck. 2021. The 'most serious' security breach ever is unfolding right now. Here's what you need to know. en. Section: Technology. (Dec. 2021). https://www.washingtonpost.com/technology/2021/12/20/log4j-hack-vulnerability-java/.

[21] [n. d.] Known Exploited Vulnerabilities Catalog | CISA. en. (). https://www.cisa.gov/known-exploited-vulnerabilities-catalog.

[22] Jake Kouns. 2008. Open Source Vulnerability Database Project. *Open Source Business Resource*, June 2008.

[23] Frank Li and Vern Paxson. 2017. A Large-Scale Empirical Study of Security Patches. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (CCS '17). Association for Computing Machinery, New York, NY, USA, (Oct. 2017), 2201–2215. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134072.

[24] Daiping Liu, Shuai Hao, and Haining Wang. 2016. All Your DNS Records Point to Us: Understanding the Security Threats of Dangling DNS Records. en. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Vienna Austria, (Oct. 2016), 1414–1425. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978387.

[25] Robert A. Martin. 2007. Common weakness enumeration. *Mitre Corporation*, 24.

[26] [n. d.] The Merit Network, Inc. ORION. en-US. https://www.merit.edu. ().

[27] [n. d.] Metasploit | Penetration Testing Software, Pen Testing Security. en. (). https://www.metasploit.com/.

[28] Kathleen Metrick, Jared Semrau, and Shambavi Sadayappan. 2020. Think Fast: Time Between Disclosure, Patch Release and Vulnerability Exploitation — Intelligence for Vulnerability Management, Part Two. en. (Apr. 2020). https://www.mandiant.com/resources/blog/time-between-disclosure-patch-release-and-vulnerability-exploitation.

[29] [n. d.] Microsoft – Security Update Guide FAQs. en-us. (). https://www.microsoft.com/en-us/msrc/faqs-security-update-guide.

[30] Aleksandar Milenkoski, Marco Vieira, Samuel Kounev, Alberto Avritzer, and Bryan D. Payne. 2015. Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices. en. *ACM Computing Surveys*, 48, 1, (Sept. 2015), 1–41. DOI: 10.1145/2808691.

[31] [n. d.] NVD - Home. (). https://nvd.nist.gov/.

[32] [n. d.] Offensive Security's Exploit Database Archive. en. (). https://www.exploit-db.com/.

[33] [n. d.] Official Snort Ruleset covering the most emerging threats. (). https://www.snort.org/products.

[34] [n. d.] Packet Storm. (). https://packetstormsecurity.com/.

[35] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. 2004. Characteristics of internet background radiation. en. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement - IMC '04*. ACM Press, Taormina, Sicily, Italy, 27. ISBN: 978-1-58113-821-4. DOI: 10.1145/1028788.1028794.

[36] Eric Pauley, Paul Barford, and Patrick McDaniel. 2023. DScope: A Cloud-Native Internet Telescope. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security 2023)*. USENIX Association, Anaheim, CA, (Aug. 2023).

[37] Eric Pauley and Patrick McDaniel. 2023. Understanding the Ethical Frameworks of Internet Measurement Studies. In *The 2nd International Workshop on Ethics in Computer Security (EthiCS 2023)*. San Diego, CA, (Feb. 2023). DOI: 10.14722/ethics.2023.239547.

[38] Eric Pauley, Ryan Sheatsley, Blaine Hoak, Quinn Burke, Yohan Beugin, and Patrick McDaniel. 2022. Measuring and Mitigating the Risk of IP Reuse on Public Clouds. English. In *2022 IEEE Symposium on Security and Privacy (SP)*. ISSN: 2375-1207. IEEE Computer Society, (Apr. 2022), 1523–1523. ISBN: 978-1-66541-316-9. DOI: 10.1109/SP46214.2022.00094.

[39] Prashanth Rajivan, Efrat Aharonov-Majar, and Cleotilde Gonzalez. 2020. Update now or later? Effects of experience, cost, and risk preference on update decisions. *Journal of Cybersecurity*, 6, 1, tyaa002. Publisher: Oxford University Press.

[40] 2023. Report a security or privacy vulnerability. en. (Jan. 2023). https://support.apple.com/en-us/HT201220.

[41] Martin Roesch. 1999. Snort – Lightweight Intrusion Detection for Networks. en, 11.

[42] Muhammad Shahzad, Muhammad Zubair Shafiq, and Alex X. Liu. 2012. A large scale exploratory analysis of software vulnerability life cycles. In *Proceedings of the 34th International Conference on Software Engineering* (ICSE '12). IEEE Press, Zurich, Switzerland, (June 2012), 771–781. ISBN: 978-1-4673-1067-3.

[43] Kiran Sridhar, Allen Householder, Jonathan Spring, and Daniel W. Woods. 2021. Cybersecurity Information Sharing: Analysing an Email Corpus of Coordinated Vulnerability Disclosure. In *The 20th Annual Workshop on the Economics of Information Security*.

[44] Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitraș. 2022. Expected Exploitability: Predicting the Development of Functional Vulnerability Exploits. en. In 377–394. ISBN: 978-1-939133-31-1. https://www.usenix.org/conference/usenixsecurity22/presentation/suciu.

[45] [n. d.] Talos - Author of the Official Snort Rule Sets. (). https://www.snort.org/talos.

[46] Johannes Ullrich. [n. d.] DShield - SANS.edu Internet Storm Center. en. (). http://www.dshield.org/index_dyn.html.

[47] [n. d.] Vulnerabilities - Security Update Guide - Microsoft. (). https://msrc.microsoft.com/update-guide/vulnerability.

[48] [n. d.] Vulnerability Reports - Latest network security threats and zeroday discoveries || Cisco Talos Intelligence Group - Comprehensive Threat Intelligence. (). https://www.talosintelligence.com/vulnerability_reports.

[49] T. Walshe and A. C. Simpson. 2022. Coordinated Vulnerability Disclosure programme effectiveness: Issues and recommendations. en. *Computers & Security*, 123, (Dec. 2022), 102936. DOI: 10.1016/j.cose.2022.102936.

[50] Vinod Yegneswaran, Paul Barford, and Vern Paxson. [n. d.] Using Honeynets for Internet Situational Awareness. en.

[51] Su Zhang, Xinwen Zhang, and Xinming Ou. 2014. After we knew it: empirical study and modeling of cost-effectiveness of exploiting prevalent known vulnerabilities across IaaS cloud. In *Proceedings of the 9th ACM symposium on Information, computer and communications security* (ASIA CCS '14). Association for Computing Machinery, New York, NY, USA, (June 2014), 317–328. ISBN: 978-1-4503-2800-5. DOI: 10.1145/2590296.2590300.

A ETHICS

Because our work measures the behavior of malicious Internet scanners, as well as publicly available repositories such as NVD, it is explicitly not human subjects research under HHS guidelines [17] and therefore falls out of scope of our institutional governance structures. With that said, any interactive measurement and data acquisition could have negative impacts on production systems, and analysis of vulnerabilities could lead to disclosure of vulnerable parties (e.g., scanner source IPs signal infected machines). We took steps in experimental design and analysis to mitigate such risks.

A.1 Traffic Collection

DScope is designed to operate with minimal interactivity to collect study endpoint data, and its behavior is a subset of what might be expected from benign services hosted on cloud IPs. While some data (such as IP addresses) could be used in conjunction with other sources to identify individuals, we did not perform such analysis and ensure the published paper does not contain such identifiable information. In addition to factors of DScope's design that mitigate personal data collection, we treat all data collected using DScope as sensitive and implement controls at the institutional level to prevent unauthorized access.

We also considered several other factors in the design of DScope to minimize potential harms:

(1) *Impact on clients.* DScope performs TCP handshakes but does not send application-layer payloads. In this sense it emulates a service that is unresponsive at the application layer, a plausible scenario under benign behavior.
(2) *Preventing IP Starvation.* DScope limits allocation of new IP addresses to minimize impact on the AWS IP pool.
(3) *Preventing compute starvation.* DScope uses preemptible (spot) instances that are reclaimable by AWS for capacity reasons, preventing impact on other tenants.
(4) *Network amplification.* We evaluated DScope to ensure that it does not amplify traffic for use in DDoS attacks.

Ethical considerations of DScope's collection are discussed further in the associated paper [36].

Group	$D - P$	SID	$A - D$	Context	Match	Adaptation
A	0d 9h	58722	0d 4h	HTTP URI	jndi	
		58723	-0d 6h	HTTP Header	jndi	
		58724	0d 22h	HTTP Header	lower	
		58725	105d 5h	HTTP URI	lower	
		58727	4d 14h	HTTP Body	jndi	
		58731	8d 21h	HTTP Header	upper	
		300057	21d 10h	HTTP Cookie	jndi	
B	0d 17h	58738	11d 7h	HTTP Header	upper	Escape sequence for $
		58739	8d 12h	HTTP Header	lower	Escape sequence for $
C	1d 15h	58741	136d 16h	HTTP Body	jndi	Escape sequence for jndi
		58742	5d 0h	HTTP Header	jndi	Escape sequence for jndi
		58744	4d 19h	HTTP URI	jndi	Escape sequence for jndi
		300058	5d 0h	HTTP Cookie	jndi	Escape sequence for jndi
D	3d 11h	58751	-3d 8h	SMTP	jndi/lower/upper	Extraneous ignored text before jndi
E	90d 3h	59246	-88d 22h	HTTP Request Method	jndi	

Table 6: Log4Shell Mitigation Variants. IDS Signature sets reveal increasingly-sophisticated attempts by attackers to thwart defenses and exploit more niche services.

A.2 Other Data Sources

All other data sources used in this work are publicly available. In some cases, this data was not in a consolidated or machine-readable format (e.g., announcements of Snort rule availability). In these cases, web pages were crawled and parsed to obtain relevant information. This crawling was performed with low rate to reduce platform impact, and complied with all relevant licenses, terms of service, and service limits (e.g., rate limiting).

B LOG4SHELL EXPLOIT VARIANTS

Table 6 contains information on the Log4Shell variants detected by Cisco IDS signature groups. "SID" refers to the Snort signature ID. "Match" refers to the Log4J command injection targeted, and "Adaptation" reflects additional adversarial adaptations that are addressed in the signature.

C VALIDATING ON ATLASSIAN CONFLUENCE (CVE-2022-26134)

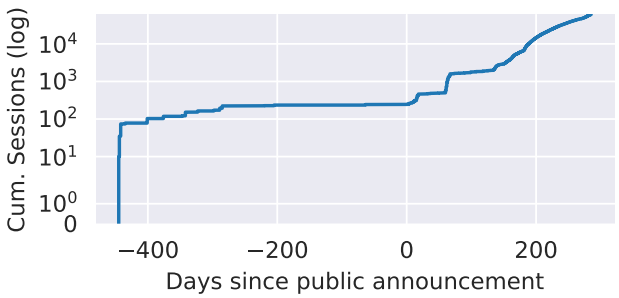


Figure 12: CDF of CVE-2022-26134 targeted TCP sessions over time.

Background. During the study period, multiple CVEs were published regarding vulnerabilities in Atlassian's Confluence product. These CVEs related to injection vulnerabilities against the Open Graph Navigation Library (OGNL), wherein an adversary could cause undesired parsing of inputs in a trusted context and trigger remote code execution. While each of these critical vulnerabilities led to broad exploitation, we focus specifically on CVE-2022-26134 (the *Confluence CVE*) for its interesting properties and large effect size. The Confluence CVE, given a CVSS score of 9.8 (Critical), allows adversaries to obtain complete access to data stored within the application, but also enables arbitrary executables to be downloaded and run on the host server. It has seen extensive exploitation in the wild for cryptocurrency mining [4] and data theft [10]. The Confluence CVE consists of improper processing of OGNL escape codes in HTTP the HTTP request path, causing them to be parsed by the application in a privileged context. Scanners can craft payloads containing java source expressions and download/run arbitrary code in a privileged context.

Analysis. While existing post-mortems have shown evidence of this CVE being exploited in the aftermath of public disclosure, our approach allows us to analyze traffic leading up to CVE publication. Through this, we examine how the Confluence CVE conforms to our broader findings about the CVE lifecycle, as well as vulnerability-specific insights obtainable through our vantage point.

C.1 CVE lifecycle

We begin by examining time-series targeting of the Confluence CVE. Note that for case studies of high-rate exploits we use TCP sessions instead of exploit events to capture the large amount of initial exploitation seen by DScope. We plot the CDF of exploit TCP sessions in Figure 12.

Finding 18 - *The Confluence CVE confirms patterns of early exploitation after publication, following by steady targeting of legacy installs.* As in our broader results, Figure 12 shows an initial spike

in exploitation after initial public disclosure. Despite this, IDS-based mitigations are largely effective on post-publication instances of this CVE, with IDS defense deployment occurring only 23 h after public disclosure. As a result, an above-average 99.6% of exploit sessions are mitigated by deployed defenses. Interestingly, this CVE shows increasing rate of exploit sessions to date, implying that adversaries see value in targeting legacy software installations.

C.2 Untargeted Exploitation

In our collected data on the Confluence CVE, we additionally see traffic from the beginning of the study period that matches the associated IDS signature. In most cases in our dataset, this leading traffic was an indicator of imprecision from the IDS signature. However, during manual analysis of the Confluence CVE we did not see such evidence. This traffic did not specifically target the Confluence-specific TCP port (8090), implying that scanners were not searching for vulnerabilities in Confluence specifically. Rather, the traffic is consistent with scanning for OGNL vulnerabilities generally, of which the Confluence CVE is an instantiation.

Finding 19 - *Adversaries can trigger novel vulnerabilities through general-purpose vulnerability scanning.* Manual analysis of leading payloads confirms that, although they were not targeted at Confluence, these payloads would lead to remote code execution on vulnerable confluence instances. Because OGNL injection is not a vulnerability in OGNL itself, there was limited publicity and proactive defense against such attacks (e.g., an IDS rule). However, the common structure of OGNL across applications makes injection vulnerabilities (due to unintended OGNL processing of user inputs) highly likely to follow a common structure in practice.

Our data suggest that some exploits show transferability to other systems using similar deployments or libraries. We anticipate that future work could use this transferability to identify novel vulnerabilities to existing exploits (and variants thereof) towards securing deployed software.

Takeaways. As with Log4Shell, the Confluence CVE confirms our findings that (a) exploitation closely follows publication, and (b) rapid deployment of mitigations is effective in preventing the majority of impact from vulnerabilities. In addition, measurement of this CVE's specific timeline allows us to observe evidence of early exploitation. This phenomenon can inform future work to proactively discover and remediate vulnerabilities.

D DESIDERATA TIME DIFFERENCE CDFS

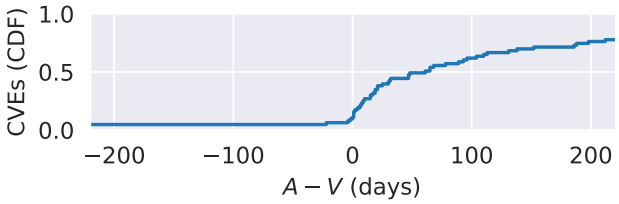


Figure 13: $A - V$ **(**$P(V \prec A = 0.90)$**).**

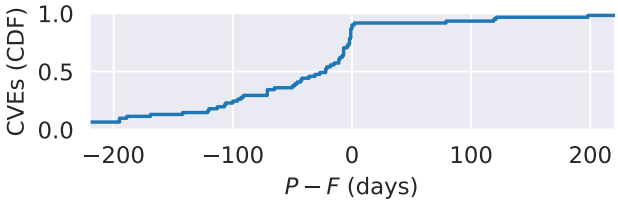


Figure 14: $P - F$ **(**$P(F \prec P = 0.13)$**).**

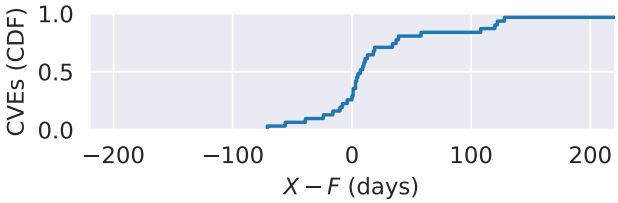


Figure 15: $X - F$ **(**$P(F \prec X = 0.74)$**).**

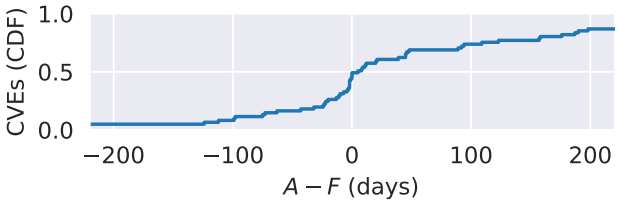


Figure 16: $A - F$ **(**$P(F \prec A = 0.56)$**).**

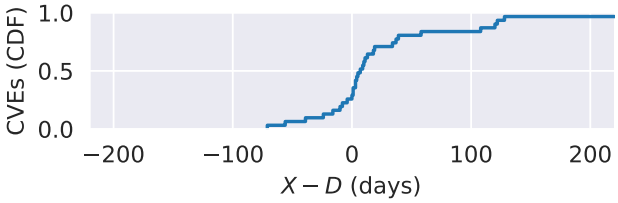


Figure 17: $X - D$ **(**$P(D \prec X = 0.74)$**).**

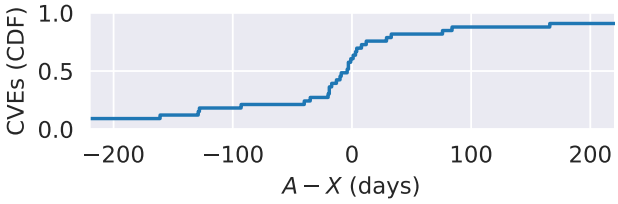


Figure 18: $A - X$ **(**$P(X \prec A = 0.39)$**).**

E TABLE OF STUDIED CVES

CVE	P	Events	Description	Impact	$D-P$	$X-P$	$A-P$	Exploit. [44]
2021-22893	2021-04-21	2	Pulse Connect Secure vulnerable URI access attempt	10.0	1d 0h	-	47d 15h	100
2021-22204	2021-04-23	16	ExifTool DjVu metadata command injection inject...	7.8	90d 12h	20d 0h	280d 22h	100
2021-29441	2021-04-27	411	Alibaba Nacos potential authentication bypass a...	9.8	168d 17h	-	263d 8h	85
2021-20090	2021-04-29	956	Arcadyan routers path traversal attempt	9.8	194d 22h	-	96d 21h	88
2021-20091	2021-04-29	19	Buffalo WSR router configuration injection attempt	8.8	194d 7h	-	352d 10h	-
2021-1497	2021-05-06	7	Cisco HyperFlex HX Installer command injection ...	9.8	0d 13h	-	188d 5h	92
2021-1498	2021-05-06	4	Cisco HyperFlex HX Data Platform command inject...	9.8	0d 13h	-	110d 3h	95
2021-31755	2021-05-07	1	Tendar Router AC11 stack buffer overflow attempt	9.8	248d 21h	-	186d 6h	92
2021-31166	2021-05-10	1	Microsoft Windows HTTP protocol stack remote co...	9.8	-	313d 0h	152d 4h	100
2021-31207	2021-05-10	15	Microsoft Exchange autodiscover server side req...	7.2	64d 17h	-	104d 5h	91
2021-32305	2021-05-18	1	WebSVN search command injection attempt	9.8	226d 15h	-	518d 12h	93
2021-21985	2021-05-26	32	VMWare vSphere Client remote code execution att...	9.8	10d 3h	50d 0h	31d 4h	99
2021-35464	2021-07-01	5	ForgeRock Open Access Manager remote code execu...	9.8	14d 12h	11d 0h	1d 21h	100
2021-21799	2021-07-16	1	TRUFFLEHUNTER TALOS-2021-1270 attack attempt	6.1	-121d 10h	1d 0h	474d 4h	99
2021-21801	2021-07-16	2	TRUFFLEHUNTER TALOS-2021-1272 attack attempt	6.1	-119d 11h	1d 0h	354d 18h	91
2021-21816	2021-07-16	4	TRUFFLEHUNTER TALOS-2021-1281 attack attempt	4.3	-79d 11h	-	165d 21h	68
2021-26085	2021-07-30	4	Atlassian Confluence information disclosure att...	5.3	410d 17h	-	68d 19h	78
2021-35395	2021-08-16	66	Realtek Jungle SDK command injection attempt	9.8	10d 13h	-	462d 22h	85
2021-26084	2021-08-26	3179	Atlassian Confluence OGNL injection remote code...	9.8	7d 12h	15d 0h	6d 6h	100
2021-40539	2021-09-07	6	Zoho ManageEngine ADSelfService Plus RestAPI au...	9.8	21d 17h	80d 0h	113d 19h	100
2021-33045	2021-09-09	29	Dahua Console Loopback potential authentication...	9.8	70d 18h	-	523d 6h	79
2021-33044	2021-09-09	34	Dahua Console NetKeyboard potential authenticat...	9.8	70d 18h	-	47d 4h	78
2021-40870	2021-09-13	2	Aviatrix Controller PHP file injection attempt	9.8	141d 14h	-	265d 11h	92
2021-38647	2021-09-15	28	Microsoft Windows Open Management Infrastructur...	9.8	6d 13h	44d 0h	4d 20h	100
2021-40438	2021-09-16	5	Apache HTTP server SSRF attempt	9.0	105d 15h	125d 0h	32d 20h	91
2021-22005	2021-09-22	5	VMware vCenter Server file upload attempt	9.8	6d 17h	16d 0h	19d 6h	100
2021-36260	2021-09-22	31117	Hikvision webLanguage command injection vulnera...	9.8	49d 21h	158d 0h	30d 4h	100
2021-39226	2021-10-05	3	Grafana authentication bypass attempt	7.3	336d 23h	329d 0h	330d 5h	55
2021-41773	2021-10-05	969	Apache HTTP Server httpd directory traversal at...	7.5	2d 13h	21d 0h	1d 2h	100
2021-27561	2021-10-15	724	Yealink Device Management server side request f...	9.8	-198d 11h	-	-220d 6h	83
2021-20837	2021-10-21	2	Movable Type CMS command injection attempt	9.8	47d 17h	9d 0h	93d 8h	91
2021-40117	2021-10-27	19074	Cisco ASA and FTD denial of service attempt	7.5	1d 12h	-	355d 11h	19
2021-41653	2021-11-13	354	TP-Link TL-WR840N EU v5 command injection attempt	9.8	30d 21h	-	8d 18h	84
2021-43798	2021-12-07	11	Grafana getPluginAssets path traversal attempt	7.5	3d 19h	15d 0h	2d 19h	100
2021-44515	2021-12-07	2	ManageEngine Desktop Central authentication byp...	9.8	35d 20h	46d 0h	212d 9h	95
2021-20038	2021-12-08	4	SonicWall SMA 100 remote unauthenticated buffer...	9.8	188d 17h	-	65d 1h	64
2021-44228	2021-12-10	6254	Apache Log4j logging remote code execution attempt	10.0	0d 19h	4d 0h	0d 13h	100
2021-45232	2021-12-27	2	Apache APISIX Dashboard authentication bypass a...	9.8	106d 19h	-	9d 17h	74
2022-21796	2022-01-28	218	TRUFFLEHUNTER TALOS-2022-1451 attack attempt	8.2	-0d 7h	-	47d 16h	61
2022-21199	2022-01-28	1	TRUFFLEHUNTER TALOS-2022-1446 attack attempt	5.9	-2d 11h	-	383d 19h	68

Continued on next page

CVE	P	Events	Description	Impact	$D-P$	$X-P$	$A-P$	Exploit. [44]
2021-45382	2022-02-17	67	D-Link router command injection attempt	9.8	112d 14h	-	1d 5h	87
2022-0543	2022-02-18	863	Debian Redis Lua sandbox escape attempt	10.0	95d 21h	40d 0h	21d 20h	100
2022-22947	2022-03-03	6	Spring Cloud Gateway Spring Expression Language...	10.0	21d 12h	150d 0h	21d 21h	100
2022-22963	2022-03-31	14	Spring Cloud Gateway Spring Expression Language...	9.8	0d 14h	1d 0h	-1d 9h	100
2022-22965	2022-04-01	107	Java ClassLoader access attempt	9.8	-	8d 0h	-387d 14h	100
2022-28219	2022-04-05	1	Zoho ManageEngine ADAudit Plus XML external ent...	9.8	92d 20h	-	138d 14h	100
2022-22954	2022-04-07	859	VMware Workspace ONE Access server side templat...	9.8	42d 17h	27d 0h	10d 17h	91
2022-29464	2022-04-18	5	WSO2 multiple products directory traversal attempt	9.8	9d 14h	11d 1h	19d 3h	100
2022-0540	2022-04-20	1	Atlassian Jira Seraph authentication bypass att...	9.8	99d 13h	-	298d 7h	94
2022-27925	2022-04-21	5	Zimbra directory traversal remote code executio...	7.2	119d 15h	-	131d 6h	100
2022-29499	2022-04-26	8	MiVoice Connect command injection attempt	9.8	70d 22h	-	61d 15h	88
2022-1388	2022-05-05	501	F5 iControl REST interface tm.util.bash invocat...	9.8	-407d 11h	8d 0h	-410d 16h	100
2022-28818	2022-05-11	7	Adobe ColdFusion cross-site scripting attempt	6.1	1d 13h	-	-299d 2h	92
2022-30525	2022-05-12	136	Zyxel Firewall command injection attempt	9.8	26d 14h	3d 0h	15d 17h	100
2022-29383	2022-05-13	1	NETGEAR ProSafe SSL VPN SQL injection attempt	9.8	41d 14h	-	198d 17h	91
2022-28958	2022-05-18	20	D-Link getcfg value command injection attempt	9.8	120d 14h	-	78d 6h	92
2022-26134	2022-06-03	50575	Atlassian Confluence OGNL expression injection ...	9.8	0d 23h	2d 0h	-444d 19h	100
2022-33891	2022-07-18	46	Apache Spark command injection attempt	8.8	17d 14h	52d 0h	17d 16h	100
2022-26138	2022-07-20	2	Atlassian Confluence hardcoded credentials use ...	9.8	6d 14h	11d 0h	15d 7h	100
2022-35914	2022-09-19	6	GLPI htmlawed php remote code execution attempt	9.8	45d 14h	36d 0h	65d 23h	100
2022-41040	2022-10-01	2	Microsoft Exchange Server remote code execution...	8.8	-0d 4h	13d 0h	89d 2h	95
2022-40684	2022-10-08	14	Fortinet FortiOS and FortiProxy authentication ...	9.8	6d 17h	10d 0h	7d 15h	100
2022-44877	2023-01-05	8	Centos Web Panel 7 unauthenticated command inje...	9.8	20d 14h	26d 0h	25d 23h	100