Historical Analysis of Exploit Availability Timelines

Allen D. Householder
CERT/CC
Software Engineering Institute
Carnegie Mellon University

Jeff Chrabaszcz
Govini

Trent Novelly
CERT/CC
Software Engineering Institute
Carnegie Mellon University

David Warren
CERT/CC
Software Engineering Institute
Carnegie Mellon University

Jonathan M. Spring
CERT/CC
Software Engineering Institute
Carnegie Mellon University

Abstract

Vulnerability management is an important cybersecurity function. Within vulnerability management, there are multiple points where knowing whether an exploit targeting a given vulnerability is publicly available would inform vulnerability mitigation priority. Despite the value of this question, there is no available historical baseline of when and how many vulnerabilities get associated public exploits. We analyze all vulnerabilities with CVE-IDs since two common repositories of public exploit data became available and find that $4.1\% \pm 0.1\%$ of CVE-IDs have public exploit code associated with them within 365 days. We analyze eight features of a CVE-ID for how they influence exploit publication. Some categories of vulnerability (CWE) are much more likely to have exploit code published than others. Vendor is a sporadic predictor of exploit publication likelihood. More vendors involved in a CVE-ID does not appear to affect exploit publication. CVSS score, commonness of the CWE, and how recently the CVE-ID was published all slightly increase the exploit publication likelihood; the confidence intervals for the size of these three effects overlap. Of 75,807 vulnerabilities studied, 3,164 had public exploits over the whole six year study; for those with exploits, the median time to publication is two days, though the mean time is 91 days.

1 Introduction

When prioritizing which vulnerabilities to respond to, whether a vulnerability is currently being actively exploited is an important feature. Predicting which exploits will become actively used would be a step better, as it would give defenders some time to patch or otherwise mitigate the vulnerability before exploitation. While predicting each and every vulnerability specifically is likely not possible, the community does not currently have access to a broad historical analysis of the usual cases for vulnerabilities and what features might indicate a vulnerability will get exploited faster than the usual case.

There are more vulnerabilities documented per year than any individual vulnerability management analyst can track. In 2019, the National Vulnerability Database (NVD)[1] added just under 20,000 vulnerabilities. Vulnerability analysts do not currently have an automated source of information about the state of exploitation of each vulnerability. Our research tools can likely be re-purposed for preliminary automated enrichment of vulnerability information.

Our research questions are shaped by this pragmatic focus. The goal of this research is to directly support vulnerability management decisions. The basic parts of vulnerability management are documented by [3]. We study exploitation because practitioners take active exploitation as "a strong indication" to respond faster [12, §6.6]. Our research goal is to provide more detail and nuance to that risk assessment by studying the historical relationship between identified vulnerabilities and public exploit code. Public exploit code is not the same as active exploitation. Active exploitation may use private code, and not all public code need be used in active exploits.

To make progress on this goal, we investigate three research questions:

RQ1 How many CVE-IDs (Common Vulnerabilities and Exposures) get public exploit code within 365 days of publication? For how many CVE-IDs is public exploit code contemporaneous with the first public documentation?

RQ2 Of those CVE-IDs that get public exploit code, how quickly is such code published? Do different sub-populations of vulnerabilities, as defined by the Common Weakness Enumeration (CWE), have different publication rates?

RQ3 Are there any features of a CVE-ID that are correlated with likelihood of public exploit code or speed of its publication?

[1]The NVD is maintained by the US Department of Commerce, National Institute of Standards and Technology (NIST). China has its own national database (CNNVD), which has a related but different set of vulnerabilities for 2019.

The rest of the paper is organized as follows. Section 2 introduces related work studying vulnerability management and prioritization. Section 3 describes data collection as well as the methodology for approaching each RQ. Section 4 presents the results, while Section 5 discusses their implications for vulnerability management.

Our study is appropriate for CSET, though we are not proposing a randomized control trial as an experiment. Our method is historiographic, and draws from medical studies on survival. If security is going to advance as a science, it will have to be able to integrate information and constraints on our knowledge into a kind of mosaic unity of different methods appropriate for different situations [23]. Vulnerability exploitation is an important but understudied facet of the cybersecurity picture, and we present this study not just for its results but also as an example of a method to study historical security data.

2 Related work

Our first priority with related work is to search for systematic review articles that contextualize our problem. Based on available reviews, exploit availability is recognized as valuable but does not appear to have been publicly measured. There is also some work in the area of vulnerability prioritization that is directly relevant that we opportunistically bring in to suggest how information about exploit availability might get used.

Searching for systematic reviews about“vulnerability management” returned only two results.[2] One only mentions vulnerability management in passing in relation to security orchestration [13]. The other is a demonstration of how CVSS (Common Vulnerability Scoring System) could be better applied to vulnerability prioritization in the industrial control system space [10]. Inspired by the precursor to [21], Figueroa-Lorenzo et al. [10] includes one database of exploit code as part of its data sources for assessment of a vulnerability, but does not provide any baseline assessment of exploit availability or any method for reliably acquiring information about exploit code for CVE-IDs.

Searching ACM CSUR instead for “CVSS” returns six more papers. Those about cloud computing are out of scope, as is game-theoretic intrusion detection. Within the information sharing literature, there has been work on the empirical and theoretical impact of vulnerability disclosure [17]. However, the state of empirical research on exploit availability as it relates to vulnerability disclosure was “a large amount of web resources contain information on available exploits” [17, p.17]. System security metrics recognize that exploit maturity is related to an increase in future attack attempts, but [19] does not indicate that exploit availability has been measured. Similarly, several metrics proposed by [5] are influenced by exploit availability (for example, attack type, attack resources, attack cost, and attack power), but exploit availability is not measured directly.

[2]All searches were conducted on Google Scholar using ‘source:CSUR’ to restrict to ACM CSUR for any article available prior to the search date of May 14, 2020.

There is some recent related work of which we are opportunistically aware. The Stakeholder-Specific Vulnerability Categorization (SSVC) [21] has an exploitation decision point which we can inform. The SSVC paper also surveys some related practitioner perspectives on the relative value of public exploit code versus other risk measures, such as CVSS. The Exploit Prediction Scoring System (EPSS) [14] relies on having information about the current state of proof-of-concept (PoC) exploit code and active exploitation before EPSS can make predictions. EPSS is far from the only attempt at using machine learning to predict exploitation (see for example [8, 9, 20]), but it has gained significant traction in the practitioner community [15].

Earlier work has answered similar but less satisfying versions of our research questions. For example, related to **RQ1** [4] appears to report 42% of vulnerabilities are exploited within 30 days, but this is out of about 100 Windows vulnerabilities targeting end-user machines running Symantec anti-virus software, and it does not have the breadth and diversity needed for our proposed historical analysis. [2] identifies exploit availability as a better indicator of patch priority than CVSS score; this motivates our work but does not address our research questions of describing properties of exploit code availability. There has also been work on the consistency, or rather inconsistency, with which security professionals assign severity to vulnerabilities [1], which suggests that more repeatable methods for assessing features of a vulnerability are warranted.

The research questions and automated data collection methodology we present here are compatible with and complementary to these efforts in the vulnerability management space.

3 Methodology

All three research questions share the same data collection and curation methodology, so we address that first in Section 3.1. Then we address the methodology for each research question in turn. The statistical methods for **RQ1** and **RQ2** are similar. For all three research questions, we base the analysis on a metaphor with survival. A CVE-ID “survives” for X days if there is not a public exploit known to target it within X days of the CVE-ID publication date.

3.1 Data

All of the data we use for our analysis is publicly available. However, it requires some effort to conveniently collect it all for analysis in one place with a consistent view. Data sources can conceptually be separated into two focus areas:

vulnerability description and publication data; and exploit code.

The common key across data sources is the CVE-ID. Therefore, our research is restricted to vulnerabilities that receive CVE-IDs. On the one hand, this analysis choice is helpful – we can exhaustively analyze the entire population of CVE-IDs, and so avoid any sampling bias. On the other hand, CVE-IDs are provided to vulnerabilities that require public coordination and tend to be given to software flaws, rather than say configuration errors [7]. To help keep these population considerations in mind during analysis, our research questions ask about CVE-IDs, not vulnerabilities in general. The vulnerabilities that get CVE-IDs are an important set of vulnerabilities, and are worth studying at least in part because they are the ones that require coordination and so consume a lot of effort by security teams. However, it is an open problem how vulnerabilities with CVE-IDs relate to vulnerabilities in general.

Vulnerability description data is collected from the following sources.

- NVD, `nvd.nist.gov/vuln/full-listing`
- CVE, `cve.mitre.org/data/downloads/index.html`
- CWE, `cwe.mitre.org/data/downloads.html`
- Common Platform Enumeration (CPE), `cpe.mitre.org/specification/`
- CERT/CC vulnerability notes, `kb.cert.org`
- TrendMicro's Zero Day Initiative (ZDI), `www.zerodayinitiative.com/advisories/published/`

Four of these sources provide the vulnerability CVE-ID and date of publication (NVD, CVE, CERT/CC, and ZDI). We treat the earliest date across these sources as the date of publication. The CERT/CC data goes back as far as 1994. However, we limit our analysis to CVE-IDs published from December 2013 because that is the first date both exploit data sources are available. The end of the analysis is January 2020, so the last vulnerability publication date analyzed is the first day of 2019.

These sources also provide some features associated with each CVE-ID, such as the CVSS (Common Vulnerability Scoring System) base score. CWE and CPE provide other associations between CVE-IDs. CWE groups CVEs into conceptual classes of weaknesses in information systems. CPE is a consistent method for identifying what type of device or software component contain the vulnerability.

We collect exploit data from the following sources.

- Exploit Database (ExploitDB), `https://github.com/offensive-security/exploitdb`
- Metasploit Framework, `https://github.com/rapid7/metasploit-framework`

These sources of public exploit code are not structured databases as the vulnerability description sources are. We use the Github repositories, rather than the web front ends for the databases, both to make it easier to automate and also so we can search developer comments and other code decoration for CVE information that is not available otherwise.

There are two challenges in mapping exploits to vulnerabilities, one structural and one logistical. The structural problem is that of identifying a minimal exploitable test case and all other test cases that are equivalent to it. This problem is studied by the fuzzing literature [18]; generally, it is accepted there is no automatic way to decide whether two exploits target the same vulnerability or to identify which vulnerability an arbitrary sample of exploit code targets. So we have to rely on human-produced notation. Thus we hit the logistical challenge of semi-structured human-produced text. The exploit data sources only care about whether an exploit sample works, not whether it is correctly tagged with a CVE-ID or if a CVE-ID exists. Exploit authors and the penetration testing companies that manage these data sources do want them to be usable, however, so we find the data quality to be adequate.

3.2 Method for RQ1

The framing of CVE-IDs "surviving" until they have associated public exploits motivates our analysis of the problem with the Kaplan-Meier estimator [16]. We use the python lifelines implementation, specifically `lifelines.KaplanMeierFitter()`.

For each CVE-ID, there are two data points, the date public and the date, if any, an exploit was published (the metaphorical date of death). We truncate times to the day, rather than treating time as a continuous variable, with all times UTC. The Kaplan-Meier estimator provides the fraction of the population of CVE-IDs that survived for at least X days, as well as a confidence interval ($\alpha = 0.05$). We have the entire population of past CVE-IDs, but we interpret the confidence interval as indicative of the range the population will continue to occupy into the future. Thus, the answer for **RQ1** is the result from the estimator after 365 days.

The second part of this research question is how many CVE-IDs survived zero days; that is, the exploit code was either the first public mention or was published on the same day as disclosure of the CVE-ID. Because we are not measuring active exploitation, this is not quite the same idea as a "zero-day exploit" because the usual understanding of zero-day is active exploitation events before the system owner or vendor is aware of the vulnerability. Our measured quantities contain neither adversary activity nor vendor awareness. The date a CVE-ID is published by a vendor is almost always well after they became privately aware of the problem. A vendor may or may not privately notify other important stakeholders before publication, as well. So CVE-IDs for which our survival value is 0 days indicate everyone knows how to exploit the CVE-ID on or before the day everyone knows about the CVE-ID. Strictly speaking, "or before" is superfluous in that definition, because publishing an exploit for a CVE-ID is a kind of vulnerability disclosure even if it is not the preferred kind. This situation is not the usual definition of zero-day, but it is certainly a situation that vulnerability managers would also like to avoid.

3.3 Method for RQ2

RQ2 has two parts, both related to the rate at which exploit code is published. The first part is the rate for the population of CVE-IDs as a whole; the second part is about if subpopulations, as defined by CWEs, have different publication rates. For each group of CWEs, we conduct the survival analysis described in Section 3.2. The exploit rate is the inverse of the survival rate . To describe the population dynamics of the speed with which CVE-IDs receive public exploits, we simply provide the descriptive statistics of the difference between `date public` and `date of exploit` for each CVE-ID – quartiles, mean and standard deviation.

Two CWEs have different survival rates after X days if the confidence intervals at that day are non-overlapping. Strictly, this intepretation of the confidence interval is if the two groups will continue to be different in the future, but it is a convenient method of representing uncertainty in CWEs that have a low sample size of public exploits.

3.4 Method for RQ3

We employ Cox proportional hazard (CPH) models [6] to determine whether a feature of CVE-IDs increases or decreases the rate at which a CVE-ID is exploited. Specifically, we use the Python `lifelines.CoxPHFitter()` implementation. This modeling choice commits us to the assumption that the different influences we want to test for can be modeled as multiplicative relationship in how they contribute to exploit occurrence. There is not prior evidence for or against this assumption in vulnerability analysis, so we adopt it tentatively.

We hypothesize the following features may influence exploitation of CVE-IDs.

- CVSSv2 base score
- CVSSv3 base score
- CWE type
- CWE size (number of CVE-IDs associated with the CWE)
- recency
- Vendor name (derived from CPE)
- CPE prefix (Vendor plus product type)
- Number of vendors involved

For each test of each feature, the method is the same. The data sources described in Section 3.1 publicly associate each of these features with CVE-IDs. Thus for each CVE-ID with a documented exploit, we check for correlated features using `fit()` with a step size of 0.25. The "recency" item is how recent the CVE-ID publication date is compared to the date of the study; that is, it tests whether exploit publication rate has been changing over time. We would expect the rate to increase if adversaries' skill accumulates faster than systems re-invent themselves [22].

vendor name	*CVE-IDs with exploits*
microsoft	100
oracle	71
debian	54
apple	51
canonical	49
linux	46
redhat	44
cisco	40
apache	38
adobe	31

Table 1: Ten vendors with the most CVE-IDs with associated public exploit code.

The result of a Cox proportional hazard model is a coefficient of relative hazard based on the group being tested. The coefficient represents a proportional increase or decrease in risk, with numbers between 0 and 1 indicating decrease (that is, multiplication by a number less than 1).

For vendor name, we required that the name have at least five CVE-IDs with published exploits to be included in the analysis. Table 1 displays the ten vendor names that are most commonly associated with published exploits.

4 Results

This section describes the results of our research questions in order. Discussion of the results is held back until Section 5.

As Figure 1 shows, the answer to **RQ1** is $4.1 \pm 0.1\%$ of CVE-IDs published between 2013 and 2019 had an associated public exploit within 365 days. Over the entire course of the study, 3,164 out of 75,807 CVE-IDs, or 4.2%, had associated public exploits. Of these 3,164 with public exploits, 1,326, or 42%, had public exploit code published on the same day the CVE-ID was publicly disclosed.

In response to **RQ2**, Figure 1 suggests that those CVE-IDs that do see public exploits tend to see them relatively quickly. Table 2 presents these results. Conditional on if a CVE-ID eventually gets an associated exploit during the study, the median time is two days. Although 75% of exploit code is observed within 28 days, there are CVE-IDs that are public for as long as six years before an exploit is published.

The rest of this section presents results for the different features examined for **RQ3**. The Cox proportional hazards model suggests that a higher CVSS score is slightly positively correlated with exploit code. The 95% confidence interval

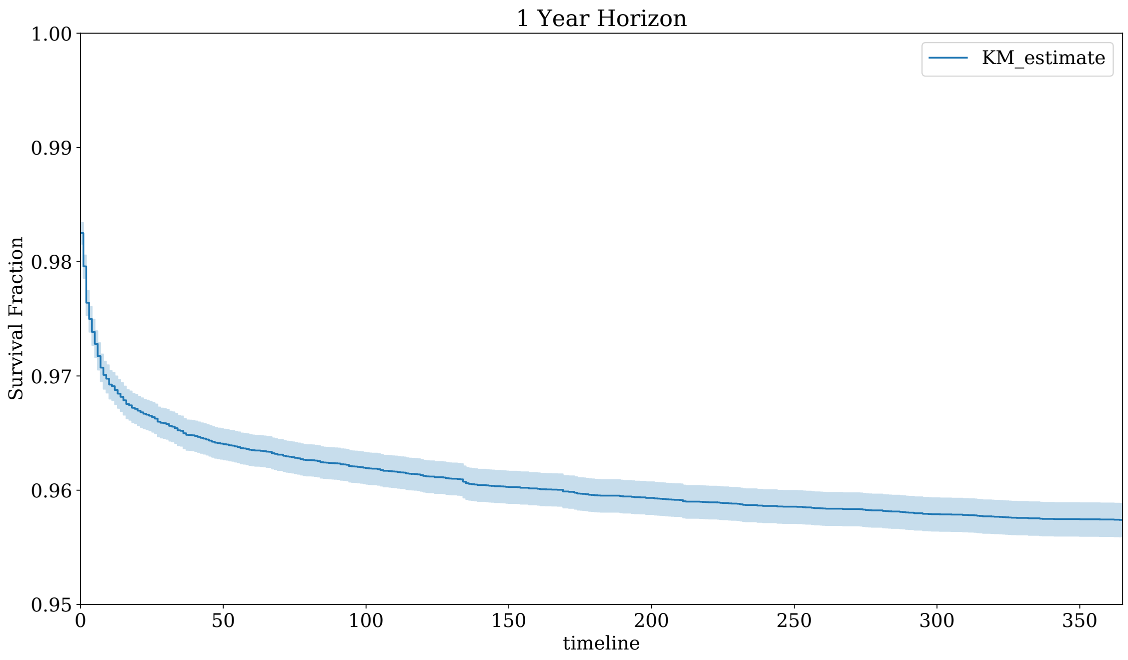

Figure 1: Fraction of CVE-IDs that have gone at least X days without an associated public exploit.

statistic	*value (days)*
min	0
first quartile	0
median	2
third quartile	28
max	2004
mean	91.5
σ	257.6

Table 2: Descriptive statistics for the speed (days) with which exploits associated with CVE-IDs are published.

CWE number	*CPH coefficient 95% CI*
CWE-20	$[1.77, 2.32]$
CWE-22	$[3.74, 5.37]$
CWE-74	$[1.22, 2.94]$
CWE-78	$[8.0, 11.25]$
CWE-89	$[6.3, 8.33]$
CWE-113	$[59.74, 91.84]$
CWE-119	$[2.39, 3.06]$
CWE-120	$[0.56, 2.27]$
CWE-125	$[0.1, 0.35]$
CWE-191	$[20.7, 40.45]$
CWE-269	$[1.51, 3.03]$
CWE-287	$[7.17, 9.78]$
CWE-294	$[0.5, 25.53]$
CWE-312	$[0.37, 5.99]$
CWE-347	$[29.67, 45.6]$
CWE-362	$[0.84, 2.41]$
CWE-367	$[3.13, 22.42]$
CWE-400	$[0.12, 0.51]$
CWE-425	$[0.97, 15.64]$
CWE-434	$[3.03, 5.7]$
CWE-476	$[0.2, 0.65]$
CWE-798	$[5.31, 9.03]$

Table 3: Results for Cox proportional hazard model intervals for selected CWE.

for the coefficient for CVSSv2 base scores is $[1.17, 1.22]$; for CVSSv3 base scores it is $[1.21, 1.30]$.

Some CWEs are highly correlated with exploit publication. There are 95 unique CWEs which are represented among the CVE-IDs for which public exploits were observed. Many of these are sparsely represented, and so we have little confidence in which direction they may impact exploit code publication. Table 3 presents a selection of the propotional hazard model results for CWEs.

CWE-362 (race condition) and CWE-367 (time-of-check to time-of-use race condition) are related; the latter is a subset of the former. The values in Table 3 for these two CWEs suggest that a general CWE category many have an uncertain

vendor name	*CPH coefficient 95% CI*
jio	$[10.18, 98.49]$
domainmod	$[8.5, 47.94]$
nuuo	$[6.05, 25.03]$
pimcore	$[2.14, 46.99]$
computrols	$[3.63, 21.76]$
alienvault	$[3.6, 14.3]$
flexense	$[2.56, 17.81]$
sophos	$[2.53, 8.33]$
zte	$[1.7, 8.17]$
nagios	$[1.77, 6.96]$
netgear	$[1.46, 6.75]$
advantech	$[1.4, 3.9]$
microsoft	$[1.34, 1.9]$
ibm	$[0.64, 0.85]$

Table 4: Results for Cox proportional hazard model intervals for vendor name from CPE for those prefixes with a p-value under 0.005.

vendor : product name	*CPH coefficient* 95% CI
domainmod:domainmod	$[8.25, 46.53]$
mcafee:virusscan_enterprise	$[6.05, 27.66]$
thekelleys:dnsmasq	$[4.31, 33.78]$
computrols:building_automation_software	$[3.56, 21.33]$
zohocorp:manageengine_servicedesk_plus	$[2.16, 34.12]$
alienvault:unified_security_management	$[2.59, 18.36]$
seeddms:seeddms	$[1.9, 24.53]$
nagios:nagios_xi	$[1.6, 10.59]$

Table 5: Results for Cox proportional hazard model intervals for vendor:product prefix from CPE for those prefixes with a p-value under 0.005.

effect on exploit publication but that certain sub-categories may significantly increase likelihood of published exploit code.

Regardless of the CWE name, as the number of CVE-IDs associated with a CWE increases the likelihood that a CVE that is a member of that CWE will have an associated public exploit increases. That is, common CWEs have proportionally more associated public exploits. The CPH 95% confidence interval for the coefficient representing the effect of CWE size is $[1.15, 1.43]$.

Recency is positively correlated with exploit publication, though the effect is relatively small. CVE-IDs published towards the end of the study are more likely to be have associated published exploits than those published at the beginning. The 95% confidence interval for the coefficient representing the effect of recency is $[1.23, 1.32]$.

number of vendors	*CPH coefficient 95% CI*
>1	$[0.85, 1.15]$
>2	$[0.9, 1.48]$
>3	$[0.9, 2.1]$
>4	$[0.61, 2.48]$
>5	$[0.86, 18.54]$
>6	$[0.05, 6.49]$
>7	$[0.17, 34.47]$

Table 6: Results for Cox proportional hazard model intervals for number of vendors affected by the CVE-ID.

Some vendors and vendor:product pairs have a reliable effect on exploit publication. Table 4 displays the vendors where the model reports a low p-value. The high variance in the confidence interval for many of these suggest that there are few observations associated with them; Microsoft and IBM are notable exceptions. Table-5 shows even fewer vendor:product pairs produce a reliable effect on exploit publication. There were 112 total vendor-product pairs, and 90 that had at least five observations. Table 6 presents the results based on more vendors being associated with the CVE-ID, regardless of who the vendors are.

5 Discussion

The answers to **RQ1** and **RQ2** are straightforward – $4.1 \pm 0.1\%$ and half of them within two days, respectively. The interpretation of the results to **RQ3** is more difficult to interpret. We first discuss the implications of the results to **RQ1** and **RQ2** before moving on to the interpretation of **RQ3**'s results.

Our results may both give vulnerability managers and cybersecurity defenders hope and fear. On the one hand, relatively few CVE-IDs have exploit code publicly available. On the other hand, for those CVE-IDs that do, it is usually public quite quickly – the median time is within two days. This result is consistent with case studies from 2011 [11] and available IDS data from 2019 [14]. Attackers do not use most vulnerabilities – because using just a few reliable ones tends to be enough for them to achieve their goals.

The recommendation for vulnerability prioritization is to heavily prioritize those vulnerabilities actively being exploited by adversaries. This is not new advice [2, 11]. Both [2, 11] warn that what exploit code adversaries actually use is not the same as what exploit code is public. But while public exploit databases are not identical with what attackers use, they are certainly an indication on what it would be trivial for attackers to use. If vulnerability managers prioritized mitigating the vulnerabilities that are trivial to exploit because code is public, our study suggests less than 4.5% of CVE-IDs need be prioritized. Our professional position is that blocking what adversaries are actively exploiting widely is the first priority, but knowing what vulnerabilities an adversary could trivially switch to and preemptively mitigating those can be equally important depending on the properties of the vulnerable system [21].

Our code for searching these exploit sources and connecting exploits with CVE-IDs is also a practical benefit to cybersecurity practitioners, who may want to know which CVE-IDs have public exploit code available and prioritize mitigating those vulnerabilities. We plan to publish this code (link redacted for blind review).

Not all vulnerabilities get CVE-IDs, and not all kinds of vulnerability are equally likely to get public exploit code. Our results demonstrate some CWEs that are more likely to see public exploit code. But there are also some kinds of vulnerabilities, such as network replay attacks, that are trivial to conduct with open-source tools (in this case, such as Nessus) [21]. Some of these may be added to Metasploit or ExploitDB as a tool config, but we have not assessed the extent to which this happens. Therefore, we expect we are slightly undercounting which CVE-IDs have public proof-of-concept exploit code.

What factors influence which CVE-IDs in particular will see exploits rather than other CVE-IDs is less clear. We have analyzed some different features than [14], but some are the same. Our perspective of the metaphor of survival analysis provides a different perspective on those features which are shared. Vendor is the most prominent shared feature. For example, both models find that CVE-IDs in Microsoft products are more likely than average to have an associated public exploit. Most incident responders probably did not need a study to tell them that, but it is nice to have intuition align with systematic results.

Unfortunately, both our study and [14] only measure correlations; causation is not highly constrained. There are some ready explanations for some of the vendors we find linked to higher chance of a public exploit. Anti-virus software and other security products are, unsurprisingly, more likely than average to have public exploit code. For vendors that specialize in security software, the effect is easier to detect. A vendor such as Microsoft is not cleanly delineated; and neither are all its products. Microsoft and Oracle may not primarily be security vendors, but they certainly develop security software that is integrated into their offerings. We hypothesize that this sort of cross-over is why vendor is not a consistent indicator of increased exploit publication risk; essentially, it is too coarse of an indicator. It is also a feature that might be unstable; mergers and acquisitions do not change the underlying software as much as they might change these labels, and sudden popularity such as Zoom experienced in March 2020 is also likely to cause discontinuities. Finally, if the end-goal of our research is to help practitioners prioritize future CVE-IDs based on past CVE-IDs, it is of limited practical use to recommend "patch all your Microsoft systems."

Some categories of vulnerability (CWE) are much more likely to have exploit code published than others. HTTP response splitting (CWE-113) is the most influential type; CVE-IDs in this category are over 60 times as likely to see exploits released. This is not entirely surprising, as web servers are both common and exposed, and this particular weakness gives the attacker a particularly high amount of control and flexibility in their attack. On the other hand, it is surprising that some CWEs are not indicative of increased risk. For example, the classic buffer overflow (CWE-120) basically has no effect. One way to interpret this result is that buffer overflow is part of the baseline from which other CVE-IDs are deviating. And perhaps, like "Microsoft," CWE-120 is so common and in such diverse applications that it is not a reliable predictor; this might indicate that it is not so much how a system is attacked but rather what value the system has that influences exploit development. For example, databases tend to be high value and we see that SQL injection (CWE-89) increases the probability of exploit publication by between 6.3 and 8.3 times. Unfortunately, we are not aware of a comprehensive metric of system value that could be used to test this hypothesis. This would be a promising direction for future work.

A large number of vendors involved in a CVE-ID has no reliable effect on the likelihood an exploit will be published. However, as only vulnerabilities that will be publicly disclosed receive CVE-IDs in the first place, we cannot rule out some confounding effects here. If many vendors are involved, a vulnerability is probably more likely to be publicly disclosed. It is unclear what effect this population change may have on the data analysis. If many vulnerabilities with few vendors are being fixed without being disclosed and given a CVE-ID, it is possible we are undercounting vulnerabilities with few vendors and how that influences the relationship between number of vendors and exploit publication depends on the properties of those invisible low-vendor-count vulnerabilities. If those vulnerabilities are systematically not seeing public exploits, then the influence of a high vendor count is higher than our study reports. However, given the high variance in the 95% confidence intervals for higher numbers of vendors, it is also possible we do not have enough observations to make reliable estimates. CVE-IDs with one vendor make up 85% of the observed exploits, and 9% have exactly two vendors. Only 0.03% CVE-IDs with vendors have over seven vendors.

CVSSv3 base score, commonness of the CWE, and how recently the CVE-ID was published all slightly increase the exploit publication likelihood; the confidence intervals for the size of these three effects overlap. The confidence intervals for these three features all include the Cox coefficient range of $[1.23, 1.30]$. This range indicates a rather small but consistent effect for all three of these features. This result does not indicate that these three features are related, only that all three are correlated with exploit publication to approximately the same degree. On the one hand, users of CVSS may find this disappointing; on the other hand, CVSS base scores are not designed to score what attackers will use and so this result is unsurprising. This result is a reminder that CVSS base scores are not, by themselves, a comprehensive hazard measure (as argued by [21], for example). Public exploit availability is also not a comprehensive hazard measure, but it clearly captures something that is not captured by CVSS base scores.

The trend that CVE-ID recency influences exploit publication is mildly worrisome. It suggests that the publication of exploit code is getting faster over time. This result may be innocuous – it may be due to improved market penetration of ExploitDB and the Metasploit Framework and more comprehensive integration of those data sources into penetration testers' workflows. However, our hypothesis is that attackers are just getting better at weaponizing vulnerabilities, either because those doing it have gotten more practice or because more people are involved. This hypothesis is consistent with the adversary capability model proposed by [22]; namely that adversary capability against any given system can only increase over time (barring major police action to arrest the actors or some such).

6 Conclusion

We have presented a historical analysis of exploit availability between 2013 and 2020. We posed three research questions with practical relevance to day-to-day vulnerability management. The first two questions, about number and speed of CVE-IDs that see exploit code published, have straightforward factual answers; $4.1 \pm 0.1\%$ and within two days are the rough answers, respectively. Our results for why this is the case, or what factors could be used to predict which CVE-IDs will fall in that $4.1 \pm 0.1\%$, are less satisfying. We find several factors correlated with exploit publication, but our research methods do not enable a clear causal explanation. Within these correlations, we find that even coarse measures such as vendor or CWE are often more highly correlated with exploit publication than the existing vulnerability severity measures represented by CVSS base scores. We hypothesize that many of the highly correlated values may be a proxy for the value of vulnerable systems; this is a question to investigate in future work.

Acknowledgments

Copyright 2020 Carnegie Mellon University. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute. NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT. [DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution. CERT Coordination Center® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Availability

We plan to make our code for searching ExploitDB and the Metasploit Framework available.

References

[1] Luca Allodi, Marco Cremonini, Fabio Massacci, and Woohyun Shim. The effect of security education and expertise on security assessments: the case of software vulnerabilities. In *Workshop on Economics of Information Security*, Innsbruck, Austria, June 2018.

[2] Luca Allodi and Fabio Massacci. Comparing vulnerability severity and exploits using case-control studies. *Transactions on Information and System Security*, 17(1):1–20, 2014.

[3] Vilius Benetis, Olivier Caleff, Cristine Hoepers, Angela Horneman, Allen Householder, Klaus-Peter Kossakowski, Art Manion, Amanda Mullens, Samuel Perl, Daniel Roethlisberger, Sigitas Rokas, Mary Rossell, Robin M. Ruefle, D'esir'ee Sacher, Krassimir T. Tzvetanov, and Mark Zajicek. Computer security incident response team (CSIRT) services framework. Technical Report ver. 2, FIRST, Cary, NC, USA, July 2019.

[4] Leyla Bilge and Tudor Dumitraş. Before we knew it: an empirical study of zero-day attacks in the real world. In *Computer and communications security*, pages 833–844, Raleigh, NC, USA, 2012. ACM.

[5] Jin-Hee Cho, Shouhuai Xu, Patrick M. Hurley, Matthew Mackay, Trevor Benjamin, and Mark Beau-

mont. STRAM: Measuring the trustworthiness of computer-based systems. *ACM Comput. Surv.*, 51(6), February 2019.

[6] David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

[7] CVE Board. CVE numbering authority (CNA) rules. Technical Report ver. 3.0, MITRE, Bedford, MA, February 2020.

[8] Michel Edkrantz, Staffan Truvé, and Alan Said. Predicting vulnerability exploits in the wild. In *International Conference on Cyber Security and Cloud Computing*, pages 513–514. IEEE, 2015.

[9] Yong Fang, Yongcheng Liu, Cheng Huang, and Liang Liu. Fastembed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm. *PLOS One*, 15(2):e0228439, 2020.

[10] Santiago Figueroa-Lorenzo, Javier Añorga, and Saioa Arrizabalaga. A survey of IIoT protocols: A measure of vulnerability risk analysis based on cvss. *ACM Comput. Surv.*, 53(2), April 2020.

[11] Dan Guido. A case study of intelligence-driven defense. *IEEE security & privacy*, 9(6):67–70, 2011.

[12] Allen D. Householder, Garret Wassermann, Art Manion, and Christopher King. The CERT®guide to coordinated vulnerability disclosure. Technical Report CMU/SEI-2017-TR-022, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2020.

[13] Chadni Islam, Muhammad Ali Babar, and Surya Nepal. A multi-vocal review of security orchestration. *ACM Comput. Surv.*, 52(2), April 2019.

[14] Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Michael Roytman, and Idris Adjerid. Exploit prediction scoring system (EPSS). In *Workshop on the Economics of Information Security*, Boston, MA, June 2019.

[15] Jay Jacobs, Sasha Romanosky, and Eireann Leverett. EPSS SIG, July 2020. Accessed 2020-07-15.

[16] Edward L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481, 1958.

[17] Stefan Laube and Rainer Böhme. Strategic aspects of cyber risk information sharing. *ACM Comput. Surv.*, 50(5), November 2017.

[18] Valentin Jean Marie Manès, HyungSeok Han, Choongwoo Han, Sang Kil Cha, Manuel Egele, Edward J Schwartz, and Maverick Woo. The art, science, and engineering of fuzzing: A survey. *IEEE Transactions on Software Engineering*, 2019.

[19] Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. A survey on systems security metrics. *ACM Comput. Surv.*, 49(4), December 2016.

[20] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *USENIX Security*, pages 1041–1056, Washington, D.C., August 2015. USENIX Association.

[21] Jonathan M Spring, Eric Hatleback, Allen D. Householder, Art Manion, and Deana Shick. Prioritizing vulnerability response: A stakeholder-specific vulnerability categorization. In *Workshop on the Economics of Information Security*, Brussels, Belgium, December 2020.

[22] Jonathan M Spring, Sarah Kern, and Alec Summers. Global adversarial capability modeling. In *APWG Symposium on Electronic Crime Research (eCrime)*, Barcelona, 5 2015. IEEE.

[23] Jonathan M Spring, Tyler Moore, and David Pym. Practicing a science of security: A philosophy of science perspective. In *New Security Paradigms Workshop*, Santa Cruz, CA, USA, October 2017.