AN ANALYSIS OF THE EFFICACY OF THE CURRENT VULNERABILITY REPORTING FRAMEWORK

Emily Shull

A Capstone Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Congdon School of Supply Chain, Business Analytics, and Information Systems

University of North Carolina Wilmington

2024

Approved by

Advisory Committee

Jeff Cummings

Ron Vetter

_____

_____

_____

Geoff Stoker

TABLE OF CONTENTS

Abstract

An Analysis of The Efficacy of the Current Vulnerability Reporting Framework.
Shull, Emily, 2024. Capstone Paper, University of North Carolina Wilmington.

Over several years, researchers, cybersecurity professionals, and the like have identified an anomaly in vulnerability reporting patterns. Occasionally, a significant spike in reporting a particular vulnerability or a vulnerability type will occur. This paper further dives into known examples of this reporting anomaly. It begins with a history of the vulnerability reporting landscape and an explanation of the phenomenon, as well as reasons that this phenomenon is harmful to the field of cybersecurity. Next, there is an explanation of the methodology used to find other instances of the phenomenon. This section uses known examples identified by other researchers juxtaposed against the data set and considers potential critiques of the methodology selected. It concludes with some suggestions for improvement in the identification process in the hopes of promoting awareness of the anomaly and limiting its occurrence in the future.

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Chapter 1: Introduction

Vulnerability reporting is one of the core ways cybersecurity professionals stay up to date with flaws identified in software products. One of the most venerated avenues for vulnerability publishing is the MITRE CVE list. The CVE list is a culmination of public and private reports thoroughly vetted by organizations with related expertise (MITRE, 2023). Sometimes, spikes occur in the volume of reported CVEs. Depending on the variety of vulnerabilities or the software affected, companies can overreact and overcorrect for a superficial threat.

This paper aims to understand if there is an identifiable pattern to the phenomenon of over-reporting the same vulnerability or type. This potential phenomenon has been discussed using several different names – "Days of Many Vulnerabilities"[1], "CVE Stuffing"[2], "Researcher Bias"[3] – but they are all identifying the same behavior pattern ([1] The Cyentia Institute, 2023; [2] Gamblin, 2020; [3] Black Hat, 2013).

Some of the potential drivers for this phenomenon could include (1) misunderstanding of the presence of a vulnerability that is occurring at a base level of an application or library, (2) building up a reputation in the field of cybersecurity for reporting vulnerabilities, or (3) following the written procedures laid out by the vulnerability databases. This list of drivers is not exhaustive but provides a starting point.

The ramifications of this unidentified pattern affect publishers and professionals relying on vulnerability databases to gather accurate information about threats to technology infrastructure. Massive influxes of reports surrounding the same issue could cause resource strain to public and private companies, federal agencies, and other stakeholders involved in the cybersecurity reporting framework.

Chapter 2: Review of Literature and Analysis

History

Starting in the 1990s, vulnerability reporting to keep track of threats to software was very decentralized, including a range of coverage, such as databases hosted by organizations to email lists supported by a few people (Bugtraq Mailing List, 2021). The options offered had points of differentiation, such as the ratio of reports to vulnerabilities, the level of disclosure, the level of public access, and the depth of information surrounding the vulnerability (Black Hat, 2013). Some companies - such as Microsoft - managed vulnerability databases for their software suite.

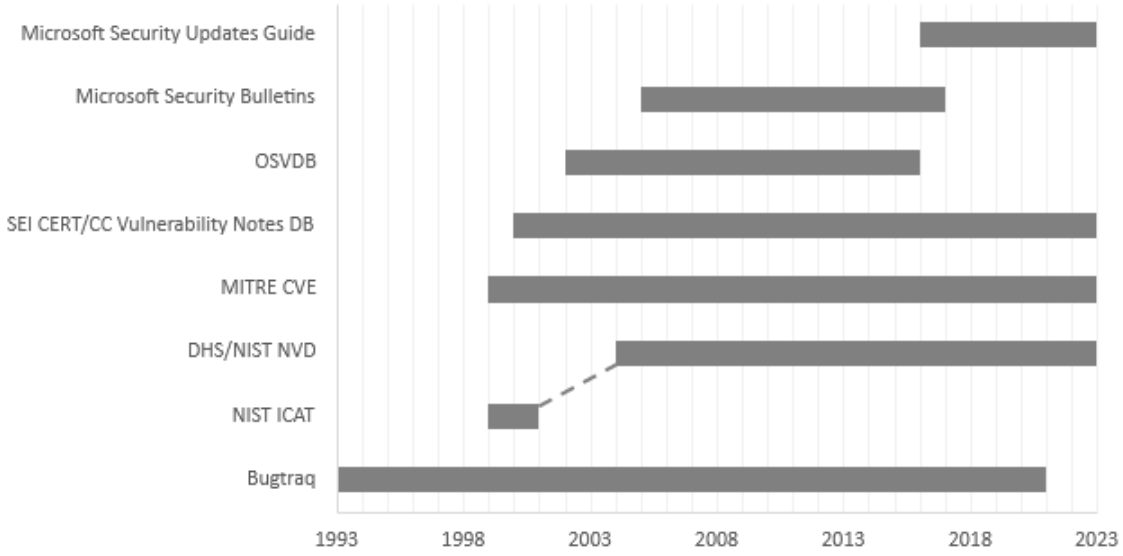

Figure 1. Vulnerability Databases by Duration (from bottom to top: Bugtraq Mailing List, 2021; The Cyentia Institute, 2023; MITRE, 2023; Vulnerability Notes Database, 2023; Gold, 2016; Microsoft, 2022; Microsoft, 2023). Note that the NIST ICAT became the basis for the DHS NVD.

However, over the next three decades, there was a consolidation within the vulnerability reporting space. Several databases failed due to lack of funding, like OSVDB[1] and NIST's ICAT[2]; others were acquired, like Bugtraq[3] ([1] Gold, 2016; [2] The Cyentia Institute, 2023; [3] Bugtraq Mailing List, 2021). In the early 2000s, two significant sources of vulnerability reporting were sponsored by the Department of Homeland Security (DHS): the MITRE CVE list[1], and NIST's ICAT, which was

renamed to the National Vulnerability Database[2] ([1] NIST, n.d.; [2] The Cyentia Institute, 2023). Although they appear to be competitors in the reporting space, the MITRE CVE list informs the National Vulnerability Database. The main distinction between the MITRE CVE list is that it seeks to publish the vulnerability with a unique identifier. In contrast, the NVD displays additional details about the vulnerability, including a scoring of the severity (Black Hat, 2013).

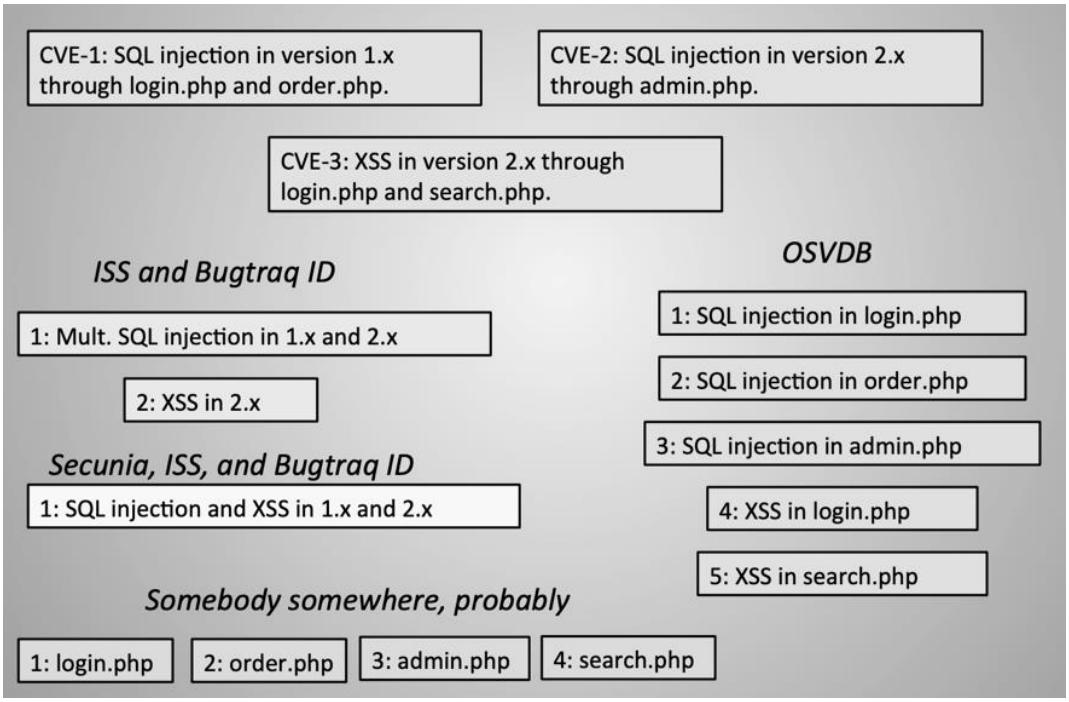

Figure 2. Vulnerability Abstraction Showing Different Focuses of VDBs (Black Hat, 2013).

The CVE ID assignment and publishing process has grown with the increasing volume of reported vulnerabilities. Due to the need for additional support to vet vulnerability reports, MITRE created the CVE Numbering Authorities (CAN) program to distribute some of the vulnerability assessment and CVE ID assignment processes to groups and organizations that specialize in the affected area. Per the CVE website: "Currently, there are 362 CNAs (360 CNAs and 2 CNA-LRs) from 40 countries and 1 no country affiliation participating in the CVE Program." (MITRE, 2024).

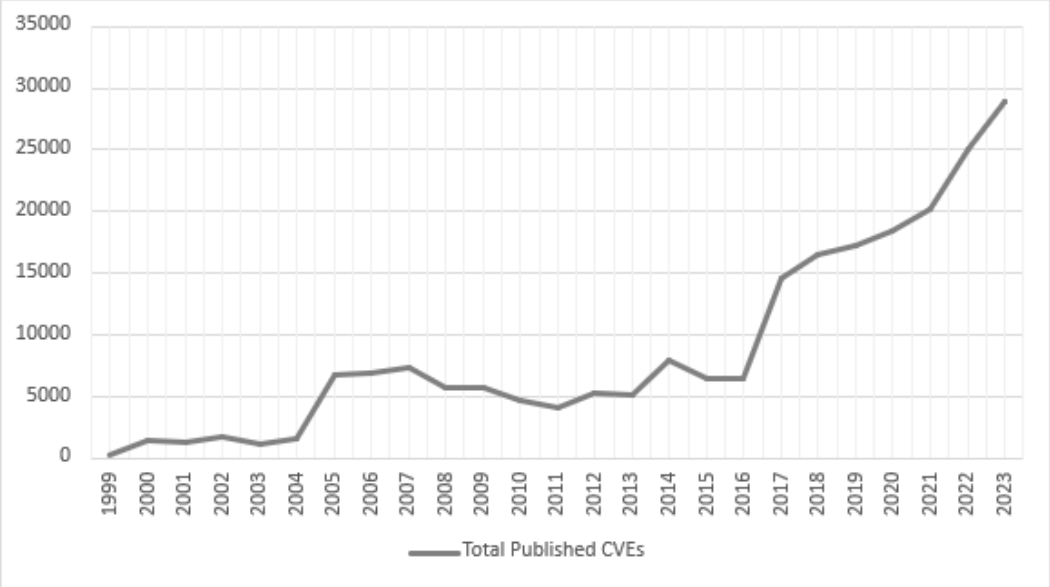

Figure 3. Total CVEs by Year (MITRE, 2024).

The CNA hierarchy includes two top-level root CNAs. Underneath the top-level CNAs, multiple Root CNAs oversee Sub-CNAs or other Root CNAs. Oversight includes training, auditing, and providing blocks of vulnerabilities to assess and publish (NIST, n.d.). Regarding the vetting process: "CNAs join the program from a variety of business sectors; there are minimal requirements, and there is no monetary fee or contract to sign" (MITRE, 2023).

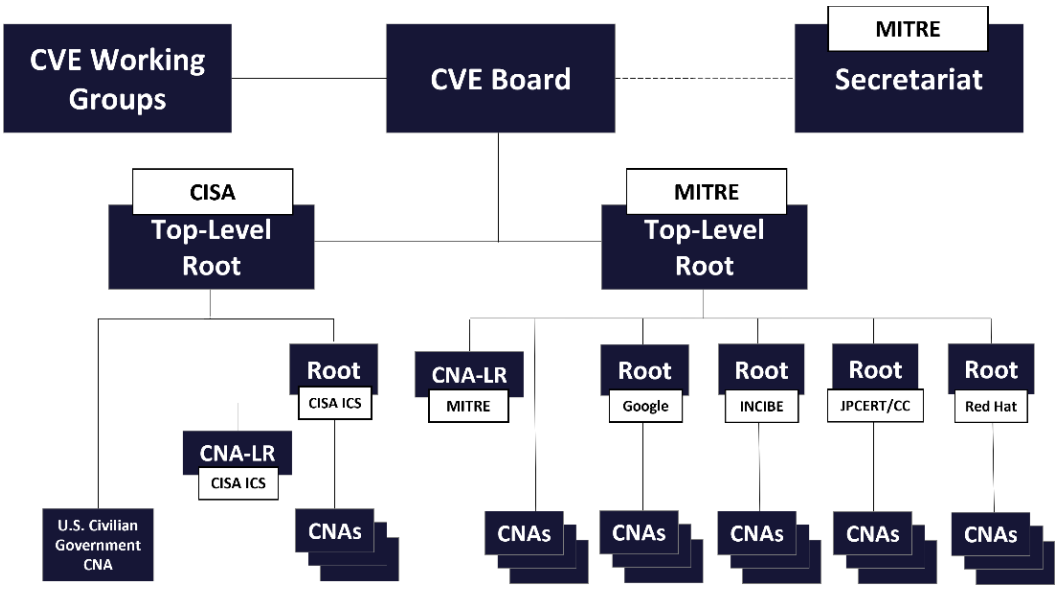

Figure 4. CNA Reporting Hierarchy (MITRE, 2024).

Literary Analysis

There have been multiple discussions surrounding the vulnerability reporting phenomenon, but they have never been consolidated because there is no uniform term to identify this pattern of behavior. One mention is a blog post by the cybersecurity professional Jerry Gamblin where he called the pattern “CVE Stuffing.” The phrase was a nod to a term used when describing denial-of-service attacks (Gamblin, Vulnerability Reporting Interview, 2023). The article cataloged an over-reporting incident for Docker in late 2020; over a week, twenty-four different CVEs were published for the same vulnerability, which had initially been recorded the year prior. The author reported several as duplicates of a previously published CVE, but “the following CVEs were filed claiming the same issue with no verification or even attempting to reach out to the container owners to let them know a CVE was filed” (Gamblin, 2020). Most of the CVEs are still open, as seen in Table 1. Gamblin explains the stakes of allowing the phenomenon to continue unnoticed: “If we get to the point where you can not (sic) even trust the data in a CVE is accurate, security teams’ ability to mitigate vulnerabilities becomes impossible” (Gamblin, 2020).

Table 1. CVEs Involved in the 2020 Docker Over-Reporting Incident

CVE ID	Status	CVSS (3.x)	CNA	Source
(Original) CVE-2019-5021	Published	9.8 (Critical)	Talos	Talos
CVE-2020-29589	Rejected (Duplicate of CVE-2019-5021)	N/A	N/A	MITRE
CVE-2020-29590	Rejected (Duplicate of CVE-2019-5021)	N/A	N/A	MITRE
CVE-2020-29591	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35184	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35185	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35186	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35187	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35188	Rejected (Duplicate of CVE-2019-5021)	N/A	N/A	MITRE
CVE-2020-35189	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35190	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35191	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35192	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35193	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35194	Rejected (Duplicate of CVE-2019-5021)	N/A	N/A	MITRE
CVE-2020-35195	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35196	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35197	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35462	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35463	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35464	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35465	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35466	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35467	Published	9.8 (Critical)	Not specified	MITRE
CVE-2020-35468	Published	9.8 (Critical)	Not specified	MITRE

List from *CVE Stuffing*, by Jerry Gamblin, 2020, https://jerrygamblin.com/2020/12/17/cve-stuffing/. Reprinted with permission.

The same behavior pattern was also referenced at the Black Hat conference in 2013 in a talk given by Steve Christey, one of the original authors of MITRE's CVE list, and Brian Martin, also known as Jericho, an influential cybersecurity professional who was at the time managing the Open-Source Vulnerability Database (OSVDB). The talk, titled "Buying into the Bias: Why Vulnerability Statistics Suck," referred to the phenomenon as research "bias." A recording of the talk provided several incidents from the earlier years of vulnerability reporting. It introduced potential reasons why the CVE list has allowed more duplicative publishing in recent years. Per the speakers, each database had a different policy for disclosure (e.g., public versus private, full versus selective) as well as a different ratio of reported vulnerabilities to published vulnerability IDs. In the case of CVE, they aimed to produce a "coordination ID" that would serve as an umbrella ID for a suite of related vulnerabilities. OSVDB, for comparison, had a 1:1 ratio of every report to a vulnerability ID (Black Hat, 2013).

With the closure of the OSVDB in 2016, it appears that the MITRE CVE list may have changed its practices regarding the ratio of IDs to reported vulnerabilities. Today, the NVD states this about the CVE ID assignment process: "If the same code affects multiple products, or multiple products are affected but with different code, a potential CVE ID will be issued for each instance" (NIST, n.d.). This policy may be a by-product of consolidating vulnerability databases in the cybersecurity landscape.

Regarding additional examples of the over-reporting phenomenon, Christey and Martin mention prolific reporters such as r0t, who "...discovered 810 vulnerabilities between Aug. 9, 2005 (sic) and Sept. 16, 2010." At the Black Hat conference, the speakers alleged that r0t was a teenager who used their school breaks to perform SQL injection and XSS attacks on a wide range of Internet sites[1]; other sources have claimed

they are a Latvian, but that no further demographic information about them could be found[2] ([1] Black Hat, 2013; [2] Algarni & Malaiya, 2013). If r0t discovered any suboptimal behavior, they would report this behavior to all the popular databases. Of note, a critique of r0t's reports is that the exploitation was usually rudimentary and out of context from the application's normal operations, so they were not comparatively as critical of threats as other more streamlined SQL injection and XSS threats (Black Hat, 2013).

A few other things mentioned in the 2013 talk have since changed. For example, Christey claimed that the MITRE CVE list publicly shows the selection criteria for vulnerabilities it must publish, including "Full-Coverage Sources...Must-Have Products...Partial-Coverage Sources" (Black Hat, 2013). There is no official reference to this type of categorization on the MITRE CVE website today[1], though being deemed a "'full-coverage source'" is mentioned in a blog post by the Zero Day Initiative (/.;'ZDI) CNA in 2020[2] ([1] MITRE, 2024; [2] Sabens, 2020).

Another term for the over-reporting phenomenon is in an article titled "The Evolving CVE Landscape" published by F5 Labs in February 2023. The article is supposed to discuss seven instances of strange CVE reports. However, before it delves into those instances it mentions a series of outlier events the author dubbed "Weird Thing 0: Days of Many Vulnerabilities" (The Cyentia Institute, 2023).

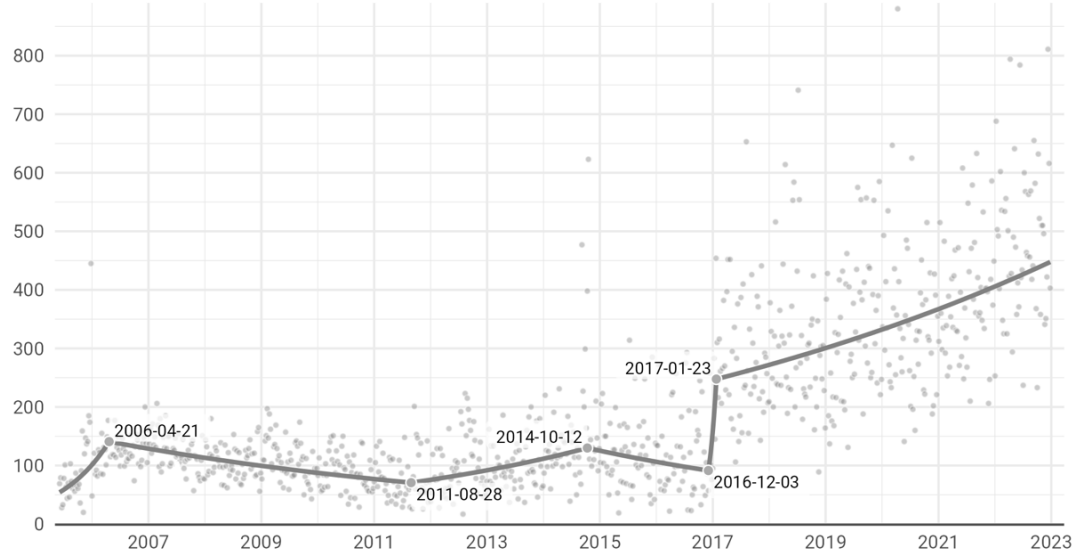

Figure 5. Number of CVEs published in NVD each week (The Cyentia Institute, 2023).

The article analyzes that these spikes are related to the over-reporting of a particular underlying vulnerability across several different application displays. It uses an incident from 2014 when multiple vulnerabilities were published surrounding verifying the Android X.509 certification (The Cyentia Institute, 2023).

When attempting to determine the root cause behind these occurrences, an interview done with the original article author, Jerry Gamblin, shed some light on potential infrastructure flaws. For example, per Gamblin, MITRE is responsible for publishing over half of all CVE IDs today rather than delegating more of that responsibility to the CNAs. Secondly, there are cases where a cybersecurity professional organically reports a vulnerability discovered and reported by the CNA responsible for that domain, which could cause some duplication if unnoticed. Lastly, there is a lot of public and private investment given to MITRE that may influence the prioritization and publishing process. Sponsors for the CVE list include the Department of Homeland Security (DHS) and the Cybersecurity and Infrastructure Security Agency (CISA), while CNAs include corporations like GitHub (MITRE, 2023).

From an individual contributor perspective, the CVE publishing rules are filled with ambiguity, which can lead to confusion when making a report. Additionally, the incentive structure of CVE reporting has become "gamified," where the number of vulnerabilities reported that have been published to the CVE list has become a way to gain reputation and clout within the cybersecurity industry (Gamblin, Vulnerability Reporting Interview, 2023).

Definition of Terms

Over the past several decades, the field of cybersecurity has created many new terms and acronyms. This section defines some terms that appear throughout this paper.

CERT/CC. CERT/CC stands for "Computer Emergency Response Team Coordination Center." Within the United States, Carnegie Mellon University (CMU) has a Software Engineering Institute (SEI) that houses a CERT/CC which maintains the Vulnerability Notes Database. According to the CERT/CC: "Vulnerability notes include summaries, technical details, remediation information, and lists of affected vendors. Most vulnerability notes are the result of private coordination and disclosure efforts" (Vulnerability Notes Database, 2023). Other countries or institutions can also have a CERT/CC; a well-known group is known as the JPCERT/CC, which is the Japan Computer Emergency Response Team Coordination Center, which acts as a Root CNA in the National Vulnerability Database (NVD) (MITRE, 2024).

CISA. CISA stands for "Cybersecurity & Infrastructure Security Agency." They are an agency within the purview of the Department of Homeland Security (DHS). The goal of this organization is to "…lead the national effort to understand, manage, and reduce risk to our cyber and physical infrastructure" (CISA, 2023). One of the critical subdivisions addresses vulnerability management. Its goal is to "Reduce the prevalence

and impact of vulnerabilities and exploitable conditions across enterprises and technologies, including through assessments and coordinated disclosure of vulnerabilities reported by trusted partners" (CISA, n.d.).

CNA. CNA stands for "CVE Numbering Authority." The definition of CNA that will be used throughout this paper comes from the MITRE CVE website: "[V]endor, researcher, open source, CERT, hosted service, bug bounty provider, and consortium organizations authorized by the CVE Program to assign CVE IDs to vulnerabilities and publish CVE Records within their own specific scopes of coverage" (MITRE, 2024). Some famous CNAs include GitHub, Inc., Siemens, and Zoom Video Communications, Inc. (MITRE, 2023).

CVE. CVE stands for "Common Vulnerabilities and Exposures" or "Common Vulnerability Enumeration." The definition of CVE that will be used throughout this capstone comes from the NIST Computer Security Resource Center (CSRC) Glossary: "A dictionary of common names for publicly known information system vulnerabilities" (NIST, n.d.).

CVSS. CVSS stands for "Common Vulnerabilities Scoring System." The definition of CVSS that will be used throughout this capstone comes from FIRST.Org, Inc.: "[It] provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity" (FIRST.Org, Inc., 2023). There are currently three different versions of the CVSS; Version 1 was superseded by Version 2 rapidly because it did not contain the necessary level of granularity to assess the vulnerability. Version 3 was introduced in 2019 and officially superseded Version 2 as of July 2022 (NIST, n.d.). Importantly, it is not a metric by which to assess the vulnerability's potential risk (NIST, n.d.).

CWE. CWE stands for "Common Weakness Enumeration." The definition of CWE that will be used throughout this paper comes from MITRE: "[A] community-developed list of software and hardware weakness types. It serves as a common language, a measuring stick for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts" (MITRE, 2023).

ICAT and *NVD.* ICAT stands for "Internet Category of Attacks Toolkit." It was started as a project by NIST and is based on the MITRE CVE database, but it provides the public with additional details about vulnerabilities. When NIST could not find enough funding in 2001, the project stalled. It was later funded by the Department of Homeland Security (DHS) and relaunched in 2005 as the National Vulnerability Database (NVD) (The Cyentia Institute, 2023).

OSVDB. OSVDB stands for "Open-Source Vulnerability Database." The OSVDB was created in 2002 but was subsequently shut down in 2016. According to HD Moore, one of its founders, the OSVDB was meant to fill a need that the founders believed would appear after acquiring the vulnerability email listserv Bugtraq. OSVDB was trying to involve the public in promoting vulnerability awareness by creating an open-source project. However, low contribution levels and the inability to procure sufficient funding to continue its mission caused the project to shut down (Gold, 2016).

Risk. The definition of risk that will be used comes from the NIST Computer Security Resource Center (CSRC) Glossary: "A measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of: (i) the adverse impacts that would arise if the circumstance or event occurs; and (ii) the likelihood of occurrence" (NIST, n.d.).

Threat. The definition comes from the NIST Computer Security Resource Center (CSRC) Glossary: “Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service” (NIST, n.d.).

Vulnerability. The definition of “vulnerability” in cybersecurity has been discussed for several years. The definition used throughout this paper comes from a University of California thesis: “…a feature or bug in a system or program which enables an attacker to bypass security measures” (Schultz, Brown, & Longstaff, 1990). Of note, NIST has a broader definition of a vulnerability beyond software flaws: “Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source” (NIST, n.d.).

Chapter 3: Methodology

The experiment aims to assess if there is a recognizable pattern of duplication within the MITRE CVE list that could indicate the frequency of reporting the same or similar vulnerabilities. In the Black Hat conference video, Martin spoke about how challenging the over-reporting weeks were for the OSVDB team. Since each vulnerability had to be vetted for accuracy, any increase beyond the expected volume of reports for a given week could quickly cause a massive diversion of resources. If VDBs were able to identify if a given published CVE was part of an over-reporting incident, or better yet, put controls into place to limit the number of occurrences of over-reporting, it would increase resource availability for initial review of reported vulnerabilities and decrease rework time of going back to mark previously published CVEs as invalid. Overall, cybersecurity professionals would be better able to assess the risk a given vulnerability poses and address only the most critical threats to their organization.

This paper attempts to understand and assess this patternicity through quantitative and qualitative means. The desired outcome of this paper is to provide recommendations for improvement to the vulnerability reporting process, such as designing a new logical control. These suggestions may be a process that vulnerability database providers can follow or an external process that consumers of the VDBs can implement. While it is unlikely that this will prevent these incidents from happening, it hopes to aid researchers and users of the databases so that they can most effectively use their limited time and resources to focus on their threat response and risk mitigation strategies for novel vulnerabilities.

Quantitative Parameters

As of early March 2024, the MITRE CVE list contains approximately 222,000 non-Rejected publishing records beginning in 1999. Due to the dynamic nature of the vulnerability reporting landscape during the late 1990s and early 2000s and considering the rapid growth in reporting volumes in recent history, only CVEs from the last decade (January 2014-February 2024) will be used. That subset measures approximately 168,000 records, showing that most vulnerabilities reported were from the previous ten years rather than the thirteen years prior.

Additionally, this date range was selected because it provides a way to verify the algorithm's validity: compare the result set against known examples of the phenomenon. The article "The Evolving CVE Landscape" mentioned the incident in 2014 for Android-based applications involving X.509 certificates, where many apps in the Google Play store were reported even though the vulnerability occurred at a lower level within the technical stack (The Cyentia Institute, 2023). Secondly, Jerry Gamblin's original blog post spoke of a 2019-2020 incident with Docker that resulted in numerous "duplicates" (Gamblin, 2020). Most recently, an incident occurred in 2023[1], where the Os Commerce product was flagged as susceptible to cross-site scripting (XSS)[2] ([1] Gamblin, Vulnerability Reporting Interview, 2023; [2] NIST, 2023). After aggregating datasets from MITRE's CVE list, an investigation can be done to see if the known incidents were flagged as incidents of over-reporting, and analysis can be performed to potentially determine a pattern of behavior indicative of an over-reporting event.

Discussion of Algorithm

The duplicates will be detected by comparing the “Description” field of one CVE within the dataset against all other subsequent CVEs. Only subsequent CVEs need to be compared because it is assumed that a preceding iteration through the dataset covered the comparison of the current CVE, thus decreasing processing time and duplicative work.

Because the detection of duplicate descriptions overlaps with the well-known scholastic concept of plagiarism, a standard plagiarism detection algorithm was selected for processing comparisons. The Jaccard index measures the similarity between two similar objects—in this case, two arrays of strings created from the breaking down of two CVE descriptions.

Sentence 1: I like dogs. I also like cats.
Sentence 2: I like to read books, but I also do not have time.

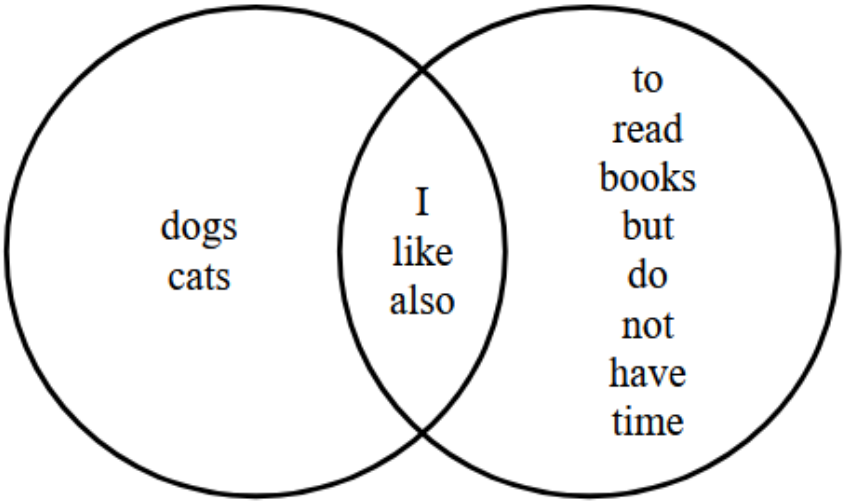

Figure 6. Illustration of the Jaccard Similarity Index. Created by the paper author.

The Jaccard index compares the elements in common within two arrays and calculates a ratio of their similarities against their differences. The similarity in the numerator of the calculation is an intersection of the two arrays; the denominator is the union minus the intersection (Costa, 2021). The result should be between zero and one, where zero is entirely dissimilar, and one is entirely identical. In the example depicted in Figure 7, the intersection of the two sentences is three: the words “I,” “like,” and “also.” The length of unique words for Sentence 1 is five (“I,” “like,” “dogs,” “also,” and “cats”), while the length of unique words for Sentence 2 is eleven (“I,” “like,” “to,” “read,” “books,” “but,” “also,” “do,” “not,” “have,” and “time”). Using the Jaccard similarity index formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

The calculation returns approximately 0.23. Therefore, the two sentences do not have that much in common.

Chapter 4: Outline of Completed Capstone

There were several stages in collecting and processing the CVE list data. In the end, with a massive number of iterations through evaluating every combination of description sets at multiple tolerance levels, a small fraction of overlapping records were returned.

Results

Three tolerance levels were assessed to discern the number of combinations of similar CVE descriptions. The levels chosen were 60%, 75%, and 90%. Levels above 50% were selected to cut out false positives due to several frequently used words comprising most of a sentence's structure.

Table 2. CVE Sets with Similarities Equal to or Greater than Tolerance Levels.

Tolerance Level	Number of Combinations Returned
60%	4,392,684
75%	1,446,251
90%	209,431

Table generated by the paper author.

Although these record volumes seem considerable, the total combinations possible, without repetition, used the following formula:

$$C_{n,k} = \binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Where *n* is the total number of records to compare; in this case, *n* would be approximately 168,000. Meanwhile, *k* represents the number of records being compared in each instance; *k* would be two for the sake of this experiment (Hetyei, 2019). This number is incalculable through usual media, such as applications like Microsoft Excel.

Within these results, several families of vulnerabilities arose. These families could represent potential over-reporting incidents. Families were grouped by how many records

in a row had the same Jaccard similarity index value. This dataset was scrubbed of outliers and aggregated into categories of matches found for depiction. Figure 7 below shows the vulnerability families by the number of matches found.

Discussion

Raw Data Collection

Beginning in 2023, the NVD officially released 2.0 API endpoints that allow users to retrieve CVE information as a JSON payload in minutes (NIST, 2022). One of the earliest identified limitations of the endpoint is that it had a record return limit of 2,000 records per callout (MITRE, 2024). A Python program was written to resolve this issue by making repeated callouts to the endpoint until less than 2,000 records were returned within a cycle. Another challenge faced when running the program involved descriptions containing special characters that were not easily translatable to an ASCII-recognized character. The solution for this problem was to use the unidecode Python library. The cleansed results were written into a JSON file for secondary processing. The total record volume from 1999 to the present was approximately 222,000 records, excluding CVE IDs with a status of "Rejected." The omission of this type was due to the "Rejected" status having a variety of assignment reasons that made it immaterial to derive value.

CVE Comparison

Once the CVE list was compiled, only records from the last ten years were selected. This dataset, comprised of 168,000 records, was chosen to utilize MITRE's contemporary reporting processes.

For this process, an algorithm was written that iterated over every CVE and compared its description to every subsequent CVE. If the Jaccard similarity index was above the provided tolerance, then the combination was written to a JSON file for

additional analysis. Results were stored in an array where the first element included the two CVE IDs, and the second element stored the Jaccard similarity index as a number with up to sixteen decimal places of precision:

["CVE-####-#####//CVE-####-#####", #.####]

Grouping and Analysis

The third step in the analysis process involved grouping the CVEs into families. Another Python program that looked for contiguous reports with the same Jaccard similarity index to group similar records was written. This solution path has some limitations, including the inability to detect related CVEs with slightly different index values and an issue with detecting if a family happens to be disrupted due to intermediate CVEs not sharing the same similarity index value.

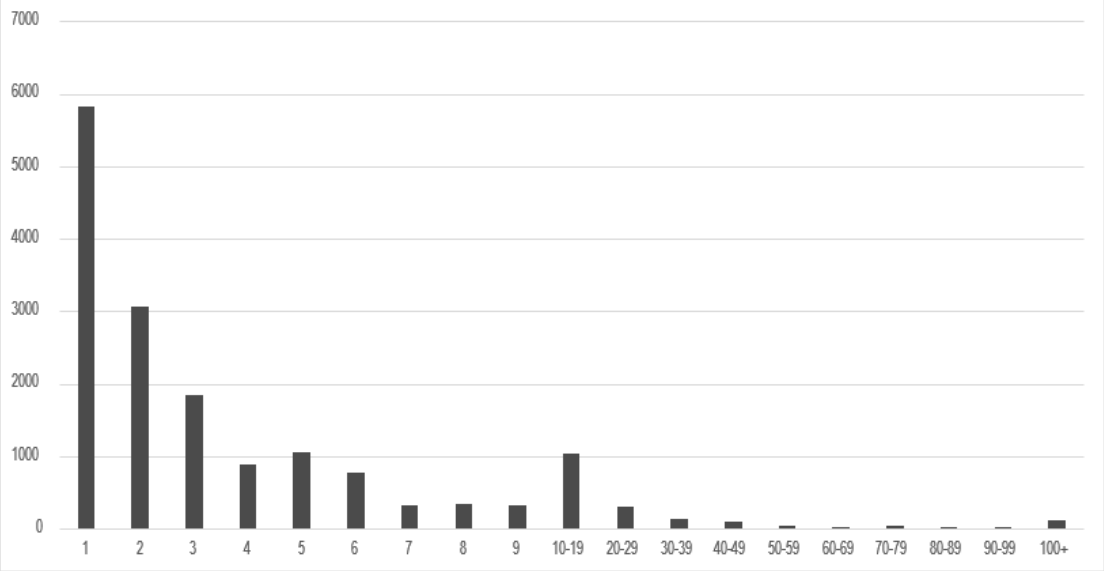

Figure 7. Vulnerability Families by Number of Contiguous Jaccard Similarity Matches. Created by the paper author.

Despite these limitations, the program results showed that vulnerability groups were frequently found close together. Two core pieces of data were excluded from the depiction in Figure 7: 26,000 records were found to have zero contiguous matches, meaning that they were one-off similarities identified. Second, in the 100+ category,

there were seven significant outliers, with a grouping size ranging from 1,544 to 36,287. Furthermore, this is an understatement for the family with 36,287, as another outlier family with 2,211 related records is for the same vulnerability report. This combined family of over 38,000 records is all related to a vulnerability surrounding an Ethereum token (NIST, 2018).

Another discovery is that several groups of vulnerabilities had a similarity value of 1.0, or complete identicalness. In these cases, the only point of differentiation was found in the "Known Affected Software Configurations" section. Potential root causes for these identical reports could be data dumps from companies into the CVE list, or an indication of poorly written descriptions from a single author attempting to mass report, or "stuff" the vulnerability report form database.

Known Incident Detection

Three incidents were previously called out as potential validation points for the software programs written to detect these incidents. The 2014 Android X.509 certificate suite of CVEs was detected at a tolerance of 60% and 75%, but not 90%.

The 2020 Docker incident reported by Jerry Gamblin was not detected at any tolerance level. This is due to the descriptions varying enough not to be considered similar by the Jaccard similarity index. This highlights a flaw in the current algorithm – it could not detect CVEs that were meaningfully similar but syntactically dissimilar. The inverse is thus also true: this program could not identify when two descriptions were syntactically similar but meaningfully dissimilar. The latter was seen most frequently when multiple command words or phrases within the same program were specified across various CVEs. The only common thread between these CVEs was that they all carried the

same Common Weakness Enumeration (CWE) ID of 306, which is for “Missing Authentication for Critical Function” (MITRE, 2024).

Lastly, the 2023 Os Commerce event was detected at all tolerance levels. The vulnerability family contained more than one hundred combinations of similarity, with an index value of 0.9375.

Chapter 5: Conclusions and Future Work

There is demonstrable evidence that work needs to be done to increase the uniqueness of reported vulnerability descriptions. There were numerous cases where similar, if not identical, descriptions were found for multiple published CVEs within a brief time. The data collected and analyzed indicates that process improvements could be identified to streamline the process and increase efficiency for MITRE, their CNAs, and those who must assess the validity of the published CVEs.

One of the most fundamental takeaways of this paper is that the over-reporting phenomenon does occur. This topic has been discussed many times over the past decades, yet the events were not linked because there is not a singular term within the industry to describe the phenomenon. A term must be decided - “CVE stuffing,” “researcher bias,” “Days of Many Vulnerabilities,” etc. - which will empower further study and improvement upon the processes that catalyzed the occurrences.

There are many reasons why over-reporting could occur; from a psychological standpoint, cybersecurity professionals may desire to gain status or reputation by stating how many vulnerabilities they reported have been published. Alternatively, from a systemic perspective, it could be a problem with how the vulnerabilities are distributed amongst CNAs or how CNAs partition the vulnerabilities within their organizational structure. This paper's purpose was not to determine why this phenomenon is occurring but to provide robust recommendations to create a more effective overall process that attempts to resolve these issues, no matter their root cause.

Suggestions for Improvement

Authorship Identification

One challenge while performing the data transformation and root cause analysis is that there is no way in the current system to trace back the original authorship of reported vulnerabilities. A unique, non-personally identifiable identifier on the vulnerability report could enable CNAs to pinpoint the beginnings of an over-reporting session. With this identifier, it could be possible for MITRE to enforce a "reports per day" limit on users and cut down on "spamming" the reporting form.

Furthermore, this identifier could provide transparency for companies looking to verify claims surrounding the number of published vulnerabilities a candidate has reported. This verification could also work at the company level, where companies could hold other companies accountable if "published CVEs" is a metric by which the company attempts to build a reputation within the industry.

MITRE Internal Policy Changes

Another potential improvement could be limiting the scope within MITRE's policies when publishing vulnerabilities. According to the National Institute of Standards in Technology (NIST), MITRE's policy for CVEs is: "If the same code affects multiple products, or multiple products are affected but with different code, a potential CVE ID will be issued for each instance" (NIST, n.d.). By supporting a policy of this nature, MITRE and its CNAs could be inflating their publishing of near-similar vulnerabilities under the premise of risk mitigation through over-disclosure, which poses a genuine problem for cybersecurity professionals attempting to assess novel vulnerabilities. For example, Appendix A shows two CVEs published by MITRE that have very similar descriptions. These descriptions are nearly identical, with only a few minor points of differentiation. Requiring cybersecurity professionals to perform this level of minute

assessment can increase confusion and decrease clarity about the surface area of the vulnerability risk.

CVE Reporting User Form Updates

When the vulnerability form is submitted to MITRE, only the following fields are required to be filled out: Vulnerability Type, which is a highly condensed choice of CWEs; Vendor of the Product(s), a freeform text entry; and the Products and Software Versions affected, also a freeform text entry (MITRE, 2024). One of the suggestions made by Jerry Gamblin during his 2023 interview was to modify the questions in the required section. Two additional questions should be added: (1) "Has the author been contacted about the vulnerability?" and (2) "What does this software do?" (Gamblin, Vulnerability Reporting Interview, 2023).

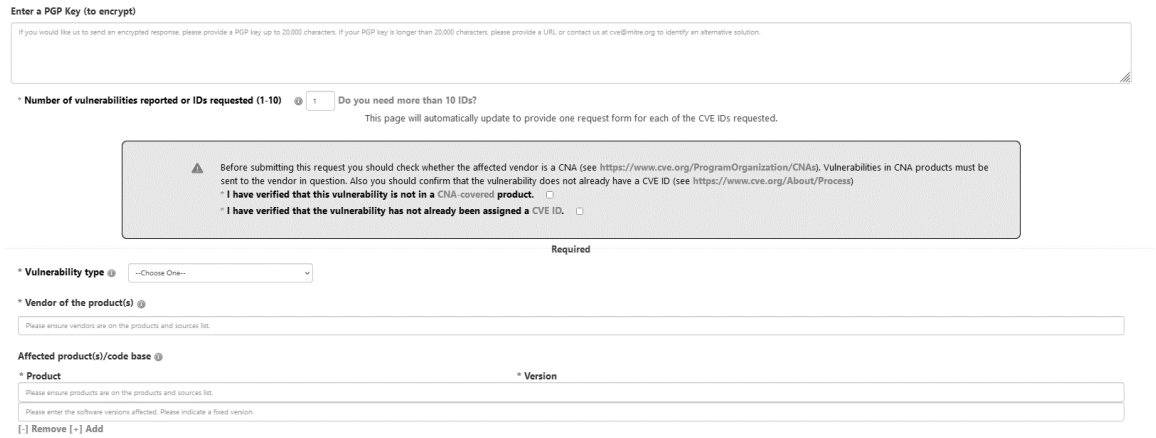

Figure 8. MITRE's Current "Submit a CVE Request" Form (MITRE, 2024).

By requesting a positive demonstration of due diligence before allowing the vulnerability to be submitted, the reporter is required to have "done their homework." Previously, reports like CVE-2023-33517 could be made for GitHub repositories that had not been actively worked on in an extended period and had no indication of an actual audience using the code (NIST, 2023).

There is a psychological benefit to this change; adding two more required fields increases the inconvenience of the user experience. This can cause outcomes such as: “If too much effort, confusion, error, and/or frustration results from attempting to engage in security-related tasks, users will simply refrain from engaging in these tasks or will perform them inadequately” (Schultz E. E., 2012). While this may seem negative, it will ideally add a layer to counteract over-reporting for personal gain. Additionally, increasing the inconvenience can have additional benefits: “Because tools and systems become convenient, while they require less time to achieve the purpose, chances of discovery that exist in the process are lost” (Kawakami, Nishimura, Katai, & Shiose, 2009).

Reports without proper verification allow for system misuse, such as boosting a professional’s vulnerability reporting numbers rather than attempting a good-faith alert of a potential weak point within a code infrastructure. In conclusion, these fields within the user form could cut down on vulnerabilities reported as “low-hanging fruit” that carry little to no impact on a broader audience.

Automated Duplication Checking

A change that could be made within MITRE’s process is to provide a period following the publication where the CVE description is run against a duplicate detection algorithm not dissimilar from the one used within this paper. There could be a 24-hour “holding” period following the publication when the soon-to-be CVE description would be compared against all CVEs published. If any are found, the CNA trying to publish the CVE would be notified of the similarities and required to either fix the description or acknowledge and force-publish the CVE.

There is a limited risk of having two similar vulnerabilities reported within the same window of time such that they are both being screened at the same time and not being screened against each other. Further testing would need to be done to understand the statistical likelihood of this occurring regularly enough to put controls in place to prevent it actively.

Future Work

Numerous related areas of study could be performed tangentially from the current paper's research. This section also discusses topics that could not be expounded upon due to the limited scope of the current paper.

One major flaw of the current research results was the differences between syntactical and meaningful similarity. This is a dissonance within computer programs now that the concentration of Natural Language Processing (NLP) within the Artificial Intelligence (AI) field is attempting to resolve. One potential avenue of study is repeating the combination comparison experiment, but this time training a model specifically designed to look for similarities in meaning. For example, the Gamblin-reported incident in 2020 with Docker could not be detected by comparison by the Jaccard similarity algorithm through syntactical comparison. Using that occurrence as a basis for training the model would show that computational complexity expands beyond mere words in common.

Also related to software development, further work could be done to refine the algorithm for grouping similar CVEs. The algorithm needs to prepare for potential interruptions in families. Additionally, in tandem with work done by the NLP, meaningfully and syntactically similar CVEs could be grouped. Lastly, considering the

differences in the “Known Affected Software Configurations” or similarities in the Weakness Enumerations could also aid in more advanced grouping methods.

An additional problem domain to be investigated is how to unify multiple reports of the same problem within a single CVE. For example, the Android incident in 2014 highlighted the issue with reporting at the application-level problems that persisted at lower levels of the technical stack. Rather than having 1,401 coexist within the NVD, they could all be consolidated into a single report that describes why all apps within the Google Play Store would be affected (MITRE, 2023). This consolidation challenge is found in many other industries, such as emergency healthcare and military operations.

As a final note, the National Vulnerability Database and MITRE’s CVE list have publicized that they know systemic issues exist within their process. During the span of this research, both websites have been under construction, with no end date specified for this work (NIST, 2024; MITRE, 2023). The National Vulnerability Database page now includes a banner: “NIST is currently working to establish a consortium to address challenges in the NVD program and develop improved tools and methods” (NIST, 2021). There are many potential avenues for improving tools and methods - hopefully to limit over-reporting incident occurrence in the future for the benefit of all stakeholders within the vulnerability reporting ecosystem.

References

Algarni, A. M., & Malaiya, Y. K. (2013). *Most Successful Vulnerability Discoverers: Motivation and Methods.* Thesis, Colorado State University, Computer Science Department, Fort Collins. Retrieved from https://www.cs.colostate.edu/~malaiya/p/SAM9766

Black Hat. (2013, December 3). *Black Hat USA 2013 - Buying into the Bias: Why Vulnerability Statistics Suck*. Retrieved March 2023, from YouTube: https://www.youtube.com/watch?v=3Sx0uJGRQ4s

Bugtraq Mailing List. (2021). Retrieved March 2023, from SecLists.Org: https://seclists.org/bugtraq/

CISA. (2023). *About CISA.* Retrieved March 2023, from Cybersecurity & Infrastructure Security Agency: https://www.cisa.gov/about

CISA. (n.d.). *Vulnerability Management*. Retrieved March 2023, from Cybersecurity & Infrastructure Security Agency: https://www.cisa.gov/vulnerability-management

Costa, L. d. (2021, November 18). Further Generalizations of the Jaccard Index. *arXiv*, 15. doi:10.48550/arXiv.2110.09619

FIRST.Org, Inc. (2023). *Common Vulnerability Scoring System SIG*. Retrieved March 2023, from FiRST: https://www.first.org/cvss/

Gamblin, J. (2020, December 17). *CVE Stuffing*. Retrieved March 2023, from JerryGamblin.com: https://jerrygamblin.com/2020/12/17/cve-stuffing/

Gamblin, J. (2023, 11 1). Vulnerability Reporting Interview. (E. Shull, Interviewer)

Gold, J. (2016, April 8). *Open-source vulnerabilities database shuts down*. Retrieved March 2023, from CSO: https://www.csoonline.com/article/3053549/open-source-vulnerabilities-database-shuts-down.html

Hetyei, G. (2019). *Permutations, combinations, and variations.* Retrieved from The University of North Carolina at Charlotte: https://webpages.charlotte.edu/ghetyei/courses/old/S19.3166/pcv.pdf

Kawakami, H., Nishimura, M., Katai, O., & Shiose, T. (2009, September). System Design based on Benefit of Inconvenience and Emotion. *ICROS-SICE International Joint Conference 2009* (p. 6). Fukuoka: ResearchGate. Retrieved from https://www.researchgate.net/profile/Hiroshi-Kawakami-6/publication/224081907_System_design_based_on_benefit_of_inconvenience_and_emotion/links/6191d2a907be5f31b781c012/System-design-based-on-benefit-of-inconvenience-and-emotion.pdf

Lily, N. H. (2017, July 1). *The Biggest Cybersecurity Disasters of 2017 So Far*. Retrieved March 2023, from Wired: https://www.wired.com/story/2017-biggest-hacks-so-far/

Microsoft. (2022, October 14). *Security Advisories*. Retrieved February 2023, from Microsoft: https://learn.microsoft.com/en-us/security-updates/securityadvisories/securityadvisories?source=recommendations

Microsoft. (2023, March). *Security Update Guide*. Retrieved March 2023, from MSRC: https://msrc.microsoft.com/update-guide/

MITRE. (2023, February 24). *Common Weakness Enumeration*. Retrieved March 2023, from CWE - Common Weakness Enumeration: https://cwe.mitre.org/

MITRE. (2023). *CVE List Downloads*. Retrieved March 2023, from CVE: https://www.cve.org/Downloads#downloads-table

MITRE. (2023). *CVE Numbering Authorities (CNAs)*. Retrieved March 2023, from CVE: https://www.cve.org/ProgramOrganization/CNAs

MITRE. (2023, July 25). *CVE Records Are Now Displayed in CVE JSON 5.0 on this Website (Updated)*. Retrieved from CVE: https://www.cve.org/Media/News/item/blog/2022/10/06/CVE-Records-Are-Now-Displayed

MITRE. (2023). *History*. Retrieved March 2023, from CVE: https://www.cve.org/About/History

MITRE. (2023). *List of Partners*. Retrieved March 2023, from CVE: https://www.cve.org/PartnerInformation/ListofPartners

MITRE. (2023). *Report/Request*. Retrieved March 2023, from CVE: https://www.cve.org/ResourcesSupport/ReportRequest

MITRE. (2023, June 25). *Search Results*. Retrieved from CVE: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Android+does+not+verify+X.509+certificates+from+SSL+servers

MITRE. (2024, March 16). *CVE Numbering Authorities (CNAs)*. Retrieved from CNAs | CVE: https://www.cve.org/ProgramOrganization/CNAs

MITRE. (2024, February 23). *CVE Services API.* Retrieved from Swagger UI: https://cveawg.mitre.org/api-docs/#/CVE%20ID/cveIdGetFiltered

MITRE. (2024, February 29). *CWE-306: Missing Authentication for Critical Function*. Retrieved from CWE: https://cwe.mitre.org/data/definitions/306.html

MITRE. (2024, March 16). *Metrics*. Retrieved March 2023, from CVE: https://www.cve.org/About/Metrics

MITRE. (2024). *Overview*. Retrieved March 2023, from CVE: https://www.cve.org/About/Overview

MITRE. (2024, March 16). *Structure*. Retrieved March 2023 , from CVE: https://www.cve.org/ProgramOrganization/Structure

MITRE. (2024, March 18). *Submit a CVE Request.* Retrieved from CVE: https://cveform.mitre.org/

NIST. (2018, August 28). *CVE-2018-13462 Detail* . Retrieved from National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2018-13462

NIST. (2021, August 19). *CVE-2021-38535 Detail* . Retrieved from National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2021-38535

NIST. (2021, August 19). *CVE-2021-38537 Detail* . Retrieved from National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2021-38537

NIST. (2022, September 20). *Change Timeline*. Retrieved from National Vulnerability Database: https://nvd.nist.gov/general/news/change-timeline

NIST. (2023, October 31). *CVE-2023-33517 Detail* . Retrieved from National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2023-33517

NIST. (2023, October 18). *CVE-2023-43703 Detail* . Retrieved from National Vulnerability Database: https://nvd.nist.gov/vuln/detail/CVE-2023-43703

NIST. (2024, March 5). *News*. Retrieved from National Vulnerability Database: https://nvd.nist.gov/general/news

NIST. (n.d.). *CNAs and CVE Counting*. Retrieved February 2023, from NVD: https://nvd.nist.gov/general/cna-counting

NIST. (n.d.). *common vulnerabilities and exposures (CVE)*. Retrieved March 2023, from Computer Security Resource Center (CSRC) Glossary: https://csrc.nist.gov/glossary/term/common_vulnerabilities_and_exposures

NIST. (n.d.). *CVEs and the NVD Process*. Retrieved March 2023, from National Vulnerability Data (NVD): https://nvd.nist.gov/general/cve-process

NIST. (n.d.). *risk*. Retrieved March 2023, from Computer Security Resource Center (CSRC) Glossary: https://csrc.nist.gov/glossary/term/risk

NIST. (n.d.). *threat*. Retrieved March 2023, from Computer Security Resource Center CSRC: https://csrc.nist.gov/glossary/term/threat

NIST. (n.d.). *vulnerability*. Retrieved March 2023, from Computer Security Resource Center (CSRC) Glossary: https://csrc.nist.gov/glossary/term/vulnerability

NIST. (n.d.). *Vulnerability Metrics*. Retrieved March 2023, from National Vulnerability Database (NVD): https://nvd.nist.gov/vuln-metrics

Sabens, S. (2020, June 22). *Our CVE Story: Bringing Our ZDI.* Retrieved from CVE Blog: https://www.cve.org/Resources/Media/Archives/Blogs/2020/2020-06-22_Our-CVE-Story-Bringing-ZDI-Community-to-CVE-Community.pdf

Schultz, E. E. (2012). Human Factors and Information Security. In *Handbook of Human Factors and Ergonomics* (Fourth ed., pp. 1250-1266). John Wiley & Sons, Inc. doi:10.1002/9781118131350.ch45

Schultz, E. E., Brown, D. S., & Longstaff, T. A. (1990, July 23). *Responding to computer security incidents: Guidelines for incident handling.* (U. o. Laboratory, Producer) Retrieved March 2023, from National Technical Reports Library (NTRL): https://ntrl.ntis.gov/NTRL/dashboard/searchResults/titleDetail/DE90016654.xhtml

The Cyentia Institute. (2023, February 28). *The Evolving CVE Landscape*. Retrieved March 2023, from F5 Labs: https://www.f5.com/labs/articles/threat-intelligence/the-evolving-cve-landscape

Vulnerability Notes Database. (2023, February 28). Retrieved March 2023, from Software Engineering Institute CERT Coordination Center: https://www.kb.cert.org/vuls/

APPENDIX A: Two CVE Descriptions Compared

CVE-2021-38535 Description

Certain NETGEAR devices are affected by stored XSS. This affects D6200 before 1.1.00.40, D7000 before 1.0.1.78, R6020 before 1.0.0.48, R6080 before 1.0.0.48, R6120 before 1.0.0.76, R6260 before 1.1.0.78, R6700v2 before 1.2.0.76, R6800 before 1.2.0.76, R6900v2 before 1.2.0.76, R6850 before 1.1.0.78, R7200 before 1.2.0.76, R7350 before 1.2.0.76, R7400 before 1.2.0.76, R7450 before 1.2.0.76, AC2100 before 1.2.0.76, AC2400 before 1.2.0.76, AC2600 before 1.2.0.76, RAX35 before 1.0.3.62, and RAX40 before 1.0.3.62.

Source: NIST, 2021.

CVE-2021-38537 Description

Certain NETGEAR devices are affected by stored XSS. This affects D6200 before 1.1.00.40, D7000 before 1.0.1.78, R6020 before 1.0.0.48, R6080 before 1.0.0.48, R6120 before 1.0.0.66, R6260 before 1.1.0.78, R6700v2 before 1.2.0.76, R6800 before 1.2.0.76, R6900v2 before 1.2.0.76, R6850 before 1.1.0.78, R7200 before 1.2.0.76, R7350 before 1.2.0.76, R7400 before 1.2.0.76, R7450 before 1.2.0.76, AC2100 before 1.2.0.76, AC2400 before 1.2.0.76, AC2600 before 1.2.0.76, and RAX40 before 1.0.3.62

Source: NIST, 2021.

APPENDIX B: GitHub Repository of CVE Analysis Algorithms

https://github.com/ewshull/UNCW_Masters_Capstone/tree/main/python_scripts