

# **Vulnerability Scoring Systems, Remediation Strategies and Taxonomies**

**by**

Jacob Friedman  
jacobfri@seas.upenn.edu

Advisor: Sebastian Angel

**EAS499 Senior Capstone Thesis**

School of Engineering and Applied Science  
Department of Computer and Information Science  
University of Pennsylvania  
May 1, 2019

## **Abstract**

This thesis focuses on vulnerability scoring systems, remediation strategies and classification systems, with the goal of improving organizational security and improving security awareness.

Section 1 will function as an introduction to the cybersecurity space, with a particular focus on business applications and introducing a number of necessary frameworks for developing security awareness. This section will also include vulnerability trend research on the largest vendors in the space and their contribution to the security problem.

Section 2 addresses cybersecurity vulnerability scoring systems and remediation strategies. There are models to detect threats and deter them, a number of them discussed in this thesis. The reality of the cybersecurity question for companies is that vulnerabilities need to be recognized, understood and fixed. To best fix vulnerabilities in a system, we first need to identify the vulnerabilities to fix and how to fix them.

Section 3 of this thesis focuses on Vulnerability Classifications and Taxonomies, strengthening understanding of threats against a system and weaknesses within a system.

Section 4 focuses on all primary topics being discussed through a different lens, as viewed by two executives in the security industry: Mike Shema, the Head of Product Security at Square, and Joe Sechman, the Head of Penetration Testing Delivery at Cobalt.

In Section 5, the thesis will conclude with insights into the disparities between industries with respect to vulnerability remediation levels and speeds, in addition to conclusions on the primary topics of the thesis. These conclusions focus on the effectiveness of scoring systems to properly capture the severity of a vulnerability and the ability of classification systems to increase security awareness. This section also offers a set of questions for future research related to the work in this thesis.

# Table of Contents

<b>1. Cybersecurity: Industry Analysis and Trends</b>	<b>4</b>
<b>1.1 Introduction to Cybersecurity</b>	<b>4</b>
<b>1.2 Cybersecurity in Business</b>	<b>5</b>
1.2.1 Security Controls & the SANS Top 20	5
1.2.2 Vulnerability detection and remediation as an NP-Hard Problem	6
<b>1.3 Vulnerability Trend Research</b>	<b>7</b>
<b>1.4 Case Study: Cobalt Vulnerability Analysis</b>	<b>8</b>
1.4.1 Uncategorized Vulnerabilities & Classification	9
1.4.2 Pen Testing Trend Analysis & Scoring	9
<b>2. Vulnerability Scoring Systems</b>	<b>13</b>
<b>2.1 Existing Vulnerability Scoring Systems</b>	<b>13</b>
2.1.1 Why do we score vulnerabilities?	13
2.1.2 CVSS Base Score	13
2.1.3 Limitations of CVSS	16
2.1.4 Bugcrowd Vulnerability Rating Taxonomy (VRT)	16
2.1.5 Cobalt Labs	17
2.1.6 Kenna Security	18
<b>2.2 Vulnerability Remediation Strategies</b>	<b>19</b>
2.2.2 CVSS Base Score	20
2.2.3 Product and Vendor Data	21
2.2.4 Category Reference Lists (on CVE ID)	22
2.2.5 Keywords and Phrases (on CVE Descriptions)	22
<b>2.3 Employee Entity Scoring Systems</b>	<b>22</b>
<b>2.4 Conclusions: Vulnerability Scoring and Prioritization Methods</b>	<b>24</b>
<b>3. Vulnerability Classifications and Taxonomies</b>	<b>26</b>
<b>3.1 Vulnerabilities, Exploits, Threats and Taxonomy Types</b>	<b>26</b>
3.1.1 Why Categorize Vulnerabilities, Threats and Exploits?	26
3.1.2 Vulnerabilities vs. Exploits	26
3.1.3 Threat Taxonomies	27
<b>3.2 Classification Standards &amp; Sources</b>	<b>29</b>
3.2.1 Common Weakness Enumeration (CWE)	29
3.2.2 Common Vulnerabilities and Exposures (CVE)	31
3.2.3 National Vulnerability Database (NVD)	33
3.2.4 OWASP Top 10	34
3.2.5 VulnCat	35
<b>3.3 Benefits of Classification Systems vs. Databases &amp; Enumerations</b>	<b>37</b>
<b>4. Views from CISOs and Security Experts</b>	<b>38</b>

<b>4.1</b>	<b>Introduction of Mike Shema</b>	<b>38</b>
<b>4.2</b>	<b>Introduction of Joe Sechman</b>	<b>38</b>
<b>4.3</b>	<b>Vulnerability Scoring</b>	<b>38</b>
<b>4.4</b>	<b>Taxonomies and Classifications</b>	<b>39</b>
<b>5.</b>	<b><i>Business Ramifications and Conclusions</i></b>	<b><i>41</i></b>
<b>5.1</b>	<b>Industry Verticals &amp; Remediation in Practice</b>	<b>41</b>
<b>5.2</b>	<b>Surprising Findings and Conclusions</b>	<b>42</b>
5.2.1	Do security teams know how to remediate?	42
5.2.2	Conclusions – Vulnerability Scoring and Classification	43
<b>5.3</b>	<b>Questions for Future Investigation</b>	<b>44</b>

# 1. Cybersecurity: Industry Analysis and Trends

## 1.1 Introduction to Cybersecurity

With security breaches increasing in frequency and significance, companies must see cybersecurity attacks as serious threats to their bottom line and public reputation. Simultaneously, consumers have become more concerned about the protection of their data and demand higher security standards from companies. Because of these two forces, effective vulnerability remediation is a growing priority in all firms.

From Target to JPMorgan Chase to Facebook, firms across all industries are feeling the repercussions of insufficient security management within their organization. Over 66% of all cybersecurity attacks against web applications are directed toward the United States, costing the average US organization over \$21 million/year, with firms like Facebook losing \$13 billion in valuation from a singular data breach [1, 12]. By 2022, the security industry is projected to exceed \$230 billion, with companies feeling pressure to dedicate multiple billions more to additional security measures. Few companies publish these budgets, but recently JP Morgan Chase announce an increase of its annual cybersecurity budget from \$250 million to nearly \$500 million [1].

Businesses' infrastructure tends to be "a compelling target for attacks, often having elevated potential for sensitive data leaks, deliberate service outages and other damage" [2]. There are a number of productive models for businesses to detect threats and deter them. Many are discussed in this thesis, but the reality of the cybersecurity question for companies is that vulnerabilities need to be recognized, understood and fixed. To best fix vulnerabilities in a system, we first need to identify the vulnerabilities to fix and how to fix them. **Section 2** of this thesis, Vulnerability Scoring and Remediation, addresses these concerns.

**Section 3** of this thesis focuses on Vulnerability Classifications and Taxonomies, attempting to strengthen understanding of threats against a system and weaknesses within a system. Most cybersecurity attacks emphasize the importance of humans as the critical linchpins in the cyberattack chain [3]. Phishing, malware and ransomware attacks make up a large portion of all successful security breaches, primarily targeted towards human actors. Improving employee understanding of security threats and different entry-points for malicious hackers can improve security awareness and limit successful human-targeted attacks. The goal of section 3 of this thesis is to analyze vulnerability classification systems and relate them to better security awareness for organizations.

**Section 4** of the thesis will bring outside perspective to the primary topics being discussed. This section will focus on security scoring systems, vulnerability taxonomies and weakness identification within an organization, as viewed by two executives in the industry:

Mike Shema, the Head of Product Security at Square, and Joe Sechman, the Head of Penetration Testing Delivery at Cobalt.

The thesis will conclude in **Section 5**, with a focus on the business ramifications of cybersecurity, its relationship with remediation strategies, vulnerability identification and classification. This section summarizes the takeaways from vulnerability scoring systems, remediation strategies, and classification systems with questions for future research.

The remaining portion of **Section 1** functions as an introduction to the cybersecurity space, starting with a particular focus on business applications and introducing a number of necessary frameworks for developing security awareness. Focus then shifts to a more granular analysis of cybersecurity: an in-depth analysis of vulnerability trends over the last 12 months. This vulnerability trend research focuses on the largest vendors in the space and their contribution to the security problem, but it also focuses on specific vulnerability datasets. These datasets should help indicate trends in types of vulnerabilities found, types of vulnerabilities exploited and where to best allocate time when it comes to improving and honing security efforts.

## 1.2 Cybersecurity in Business

### 1.2.1 Security Controls & the SANS Top 20

The SANS Institute, a private company that specializes in information security, posts a Top 20 list, focusing on security controls for effective defense. During an interview with Mike Shema (see section 4), Mike identified the SANS Top 20 as one of the primary practical controls of organizational security. All 20 of these security controls are detailed below [4]:















SANS TOP 20 CRITICAL SECURITY CONTROLS	
1. Inventory of Authorized & Unauthorized Devices 	11. Secure Configurations for Network Devices
2. Inventory of Authorized & Unauthorized Software 	12. Boundary Defense 
3. Secure Configurations for Hardware & Software on Mobile Devices, Laptops, Workstations, & Servers 	13. Data Protection 
4. Continuous Vulnerability Assessment & Remediation 	14. Controlled Access Base on the Need to Know 
5. Controlled Use of Administrative Privileges 	15. Wireless Access Control
6. Maintenance, Monitoring, & Analysis of Audit Logs 	16. Account Monitoring & Control 
7. Email and Web Browser Protections 	17. Security Skills Assessment & Appropriate Training to Fill Gaps
8. Malware Defenses 	18. Application Software Security 
9. Limitation and Control of Network Ports, Protocols, and Services 	19. Incident Response Management
10. Data Recovery Capability	20. Penetration Tests & Red Team Exercises

Figure 1: SANS Top 20 Security Controls [4]

The SANS Top 20 provides actionable and organizationally focused recommendations for security systems. Functioning as a highly developed security checklist, the SANS Top 20 Security Controls recommends anti-malware scans, which can be easily installed and performed by software services like Malwarebytes. Similarly, it recommends inventory of all devices, an essential security measure that should be part of the minimum baseline for security control. While it is not a highly-technical approach to understanding of cybersecurity defense, the SANS Top 20 functions as an attempt at business-focused security.

The goal of this thesis is not only to delve into vulnerability scoring systems, remediation strategies, taxonomies and classifications, but also how to effectively implement these concepts in an organization. This will remain a thread throughout the entire thesis, noting the lean of specific models and systems towards the academic and the application of others to business.

### 1.2.2 Vulnerability detection and remediation as an NP-Hard Problem

In a paper titled *An effective computational technique for taxonomic position of security vulnerability in software development*, Srivasta and Kumar describe the necessity of a comprehensive system where cybersecurity standards can be applied to business processes, citing vulnerability detection as an NP-Hard problem [5]. On average, software engineers spend 70-80% of their time focused on the information security risk management phase of testing. This type of work increases exponentially in time and complexity as the size and costs of a system are increased. Srivasta and Kumar describe this security remediation process using the following algorithm [5]:

1. Apply the test case for verification of software system
2. **IF** {security bugs identified during the testing by a test case} **THEN**  
The presumption is that updation is required on predecessor phase but do not have the potential position.  
    **IF** {Predecessor phase is coding} **THEN**  
        Update and repeat step 1.  
    **ELSE**  
    **IF** {Predecessor is not coding phase} **THEN**  
        Update in respective phase work and repeat the successor phase of the current phase.
3. **ELSE** Go for deployment.

It is difficult to standardize practices that may vary by organization, but this procedure is a well-generalized process for deploying security bug-free code. However, this process clearly solidifies itself as an increasingly complex problem that grows in complexity along with the system itself. The primary analysis in this thesis takes this generalized algorithm and applies effective vulnerability scoring systems and remediation strategies to minimize the compounded complexities associated with deploying secure code.

### 1.3 Vulnerability Trend Research

Before delving into specifics, it is important to address a number of industry trends and observations. These analyses primarily come from Kenna Security’s Prioritization to Prediction Reports and a dataset of vulnerabilities procured from Cobalt Labs, a crowdsourced penetration testing company [6, 8]. With a better understanding of industry trends and vulnerability trends, this thesis can approach scoring systems, remediation strategies and types of classifications with a stronger background.

The security space is highly segmented, with few security provider types proven as security “giants”. With penetration testing, vulnerability scanners, security consultancies, firewall and anti-malware providers, the space is diverse and offers significant choice for buyers. This is a productive way to maintain security, as sticking with any one provider for all security needs can be inherently less secure than many providers. Despite this, there is massive consolidation among vendors that provide software to most of the organizations in the world. Companies like Microsoft, IBM, Adobe, Oracle and Red Hat offer a wide array of products that are used nearly universally. Because of this, focusing on vulnerabilities found by specific vendors is essential in maintaining security.

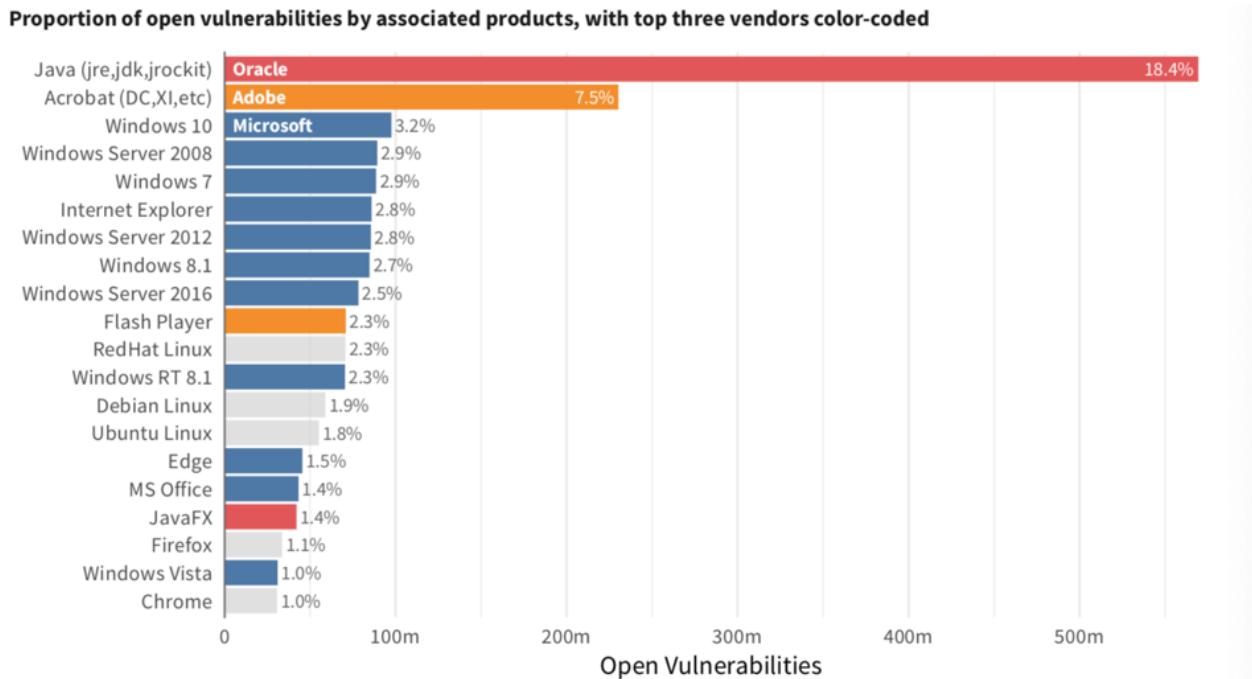


Figure 2: Proportion of open vulnerabilities by associated products [6]

Oracle, Microsoft and Adobe are together responsible for 69.1% of all open vulnerabilities observed by consumers, with 34.4%, 17.6% and 17.1% of all open vulnerabilities respectively [6]. For the sake of these statistics, Oracle encompasses all Sun Microsystems technologies and IBM includes Red Hat. When we break this down a bit further into specific products, we can see



significant discrepancies based on the offerings provided [6]. Clearly, Java has significantly more open vulnerabilities than any product offered, but we see a number of other products with large numbers of open, publicly disclosed vulnerabilities. The purpose of this section is not to criticize organizations like Oracle and Microsoft for such large swaths of vulnerabilities, but instead to indicate why these sets of vulnerabilities are so important.

Only 15.6% of all open vulnerabilities have any sort of known exploit in the wild and many organizations, particularly Microsoft, are incredibly fast at remediating these open, exploitable vulnerabilities [6]. Microsoft specifically, on average, reaches 25% of all open vulnerabilities in 14 days and 50% of all vulnerabilities in 37 days. The speed of this remediation is impressive and is fueled primarily by their commitment to weekly patches. However, an organization like IBM is one of the slowest with respect to remediation velocity (25% reach in 225 days and 50% reach in 578 days) [7]. Here, we see a different strategy when approaching vulnerabilities. Instead of focusing on speed, they focus on necessity of patches. Even more complicating is that very few consumers even install or update security patches in a timely manner. Microsoft has the strongest patch adoption speeds, but many organizations that release patches see consumers installing or updating weeks and months after releases.

It would seem intuitive that all vendors who offer products with publicly disclosed security vulnerabilities have an obligation to release fixes and patches for these weaknesses. However, this assertion is not quite correct. With such a low percentage of open vulnerabilities with known exploits (15.6%), do consumers demand patches for only this set of vulnerabilities? Or do we expect organizations like Oracle to spend years and years patching exploit-free open vulnerabilities, slowing down progress of new products simply for 100% coverage? This tradeoff is one that consumers and vendors need to consider when analyzing security concerns and how to best spend remediation efforts.

#### **1.4 Case Study: Cobalt Vulnerability Analysis**

Cobalt Labs (“Cobalt” or “Cobalt.io”) is a crowdsourced penetration testing platform, offering penetration testing for organizations looking for vulnerabilities in their applications. Cobalt’s penetration testing system is built on teams of 3-5 white-hat hackers across the globe that search for vulnerabilities and report them to clients who can remediate these vulnerabilities before they are exploited in production. Cobalt and their vulnerability scoring system are described in greater detail in section 2.1.5. This section will focus on analyzing two distinct vulnerability datasets from Cobalt.

### 1.4.1 Uncategorized Vulnerabilities & Classification

The first dataset is comprised of reported vulnerabilities discovered by white-hat hackers through the Cobalt application from July 2017 to June 2018 [8]. Cobalt identifies vulnerabilities by categorizing them using a modified version of the most recent OWASP Top 10, but a meaningful portion of the vulnerabilities discovered are not categorized under one of these primary buckets. These 10 buckets, published biennially by OWASP, attempt to focus the security industry on the 10 most commonly found vulnerability types discovered in the industry. The OWASP Top 10 is described in greater detail in section 3.2.3. During the time frame of interest, over 450 vulnerabilities fell into this uncategorized bucket, or “Other” vulnerabilities. This analysis is focused on these 450 “Other” vulnerabilities.

Taking these 450 vulnerabilities, the primary goal was to find a more comprehensive categorization system to better organize these weaknesses. CWE, the Common Weakness Enumeration, which is further discussed in Section 3.2.1, is a comprehensive list of all publicly disclosed weaknesses and membership groups for them. Using CWE, the 450 vulnerabilities fell into a whopping 86 unique CWE categories [8]. With the initial goal of trying to find commonality in these vulnerabilities, this shows the surprising variability and uniqueness of the dataset. Outside of the OWASP Top 10, the CWE mappings showed a far more diverse set of weaknesses than expected.

Some miscellaneous findings from this analysis showed a significant number of vulnerabilities classified under CWE-80 (Improper Neutralization of Script-Related HTML Tags in a Web-Page), CWE-918 (Server-Side Request Forgery), and CWE-611 (Improper Restriction of XML External Entity Reference). Three of these CWE IDs fall under SSRF, which indicates that the modified OWASP Top 10 could benefit from a dedicated SSRF bucket, decreasing the number of miscellaneously identified vulnerabilities [8].

CWE also offers “Member Of” classification of each CWE ID, from which we can find relevant buckets for each weakness. Taking the current dataset, ironically 73 of the 450 vulnerabilities are subcategorized under the OWASP Top 10, indicating that white-hat hackers may not have the strongest understanding of classification systems themselves. These types of findings, where nearly 20% of all vulnerabilities are miscategorized or poorly bucketed, will provide impetus for the analysis in Section 3.

### 1.4.2 Pen Testing Trend Analysis & Scoring

Second is an analysis of a larger dataset of reported vulnerabilities through the Cobalt application from January 2017 to June 2018 [9]. This dataset includes all vulnerabilities reported during that time span, which totals to over 6000 vulnerabilities. This dataset is comprised of a number of clients across a wide array of industries and offers an incredibly representative set of security vulnerabilities found in the “wild” over a recent 18-month period.

As described above, Cobalt categorizes these vulnerabilities with a modified version of the OWASP Top 10, with a catch-all bucket for vulnerabilities that may not seem to fit any bucket. Following the trends in the OWASP Top 10 buckets can act as a proxy for trends in specific vulnerability types in the wild and changes in the severity of them. In Figure 3, we can clearly see the growth of found vulnerabilities in each OWASP bucket, most notably finding a significant number of vulnerabilities in the Misconfiguration categorization.

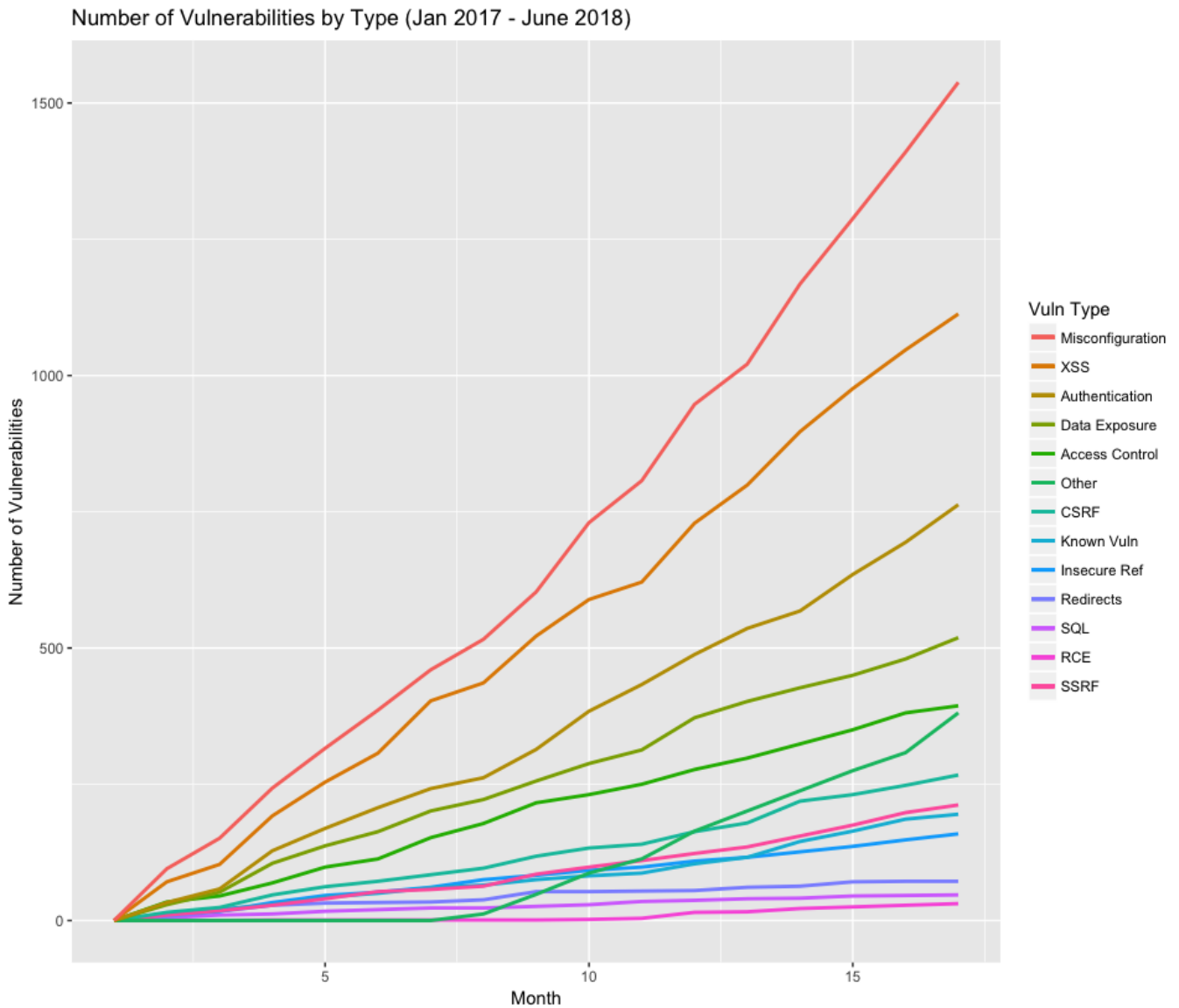


Figure 3: Number of Vulnerabilities by Type (Jan 2017 – June 2018)

This graph was developed by downloading the entire 6000+ vulnerability dataset, provided by Cobalt. The data was already anonymous and was manually cleaned and reformatted before being plotted over time, with month zero as June 2018.

Misconfiguration, or Security Misconfiguration, according to OWASP is “the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information” [10]. Recognizing these vulnerabilities as the most commonly found in the Cobalt dataset supports the claim from OWASP that Security Misconfiguration, albeit a broad bucket, is undeniably one of the most concerning security areas.

Vulnerability scoring systems will be discussed in great detail in Section 2 but analyzing the Cobalt dataset of over 6000 vulnerabilities will provide valuable background for this topic. In particular, SQL Injection and Cross Site Scripting (XSS) vulnerabilities are consistently rated as the most significant vulnerabilities found, with average criticality scores of 16.93 and 10.63 out of 25, respectively. SQL Injection is a Cobalt-narrowed version of OWASP’s Injection category, while XSS stands alone in the OWASP Top 10, indicating the prevalence of these two vulnerabilities in the security community.

Focusing more specifically on the vulnerability criticality scores in this dataset, we can see the normalized criticality scores by score bucket, ranging from 0 as the least critical vulnerability to 25 as the most critical\*:

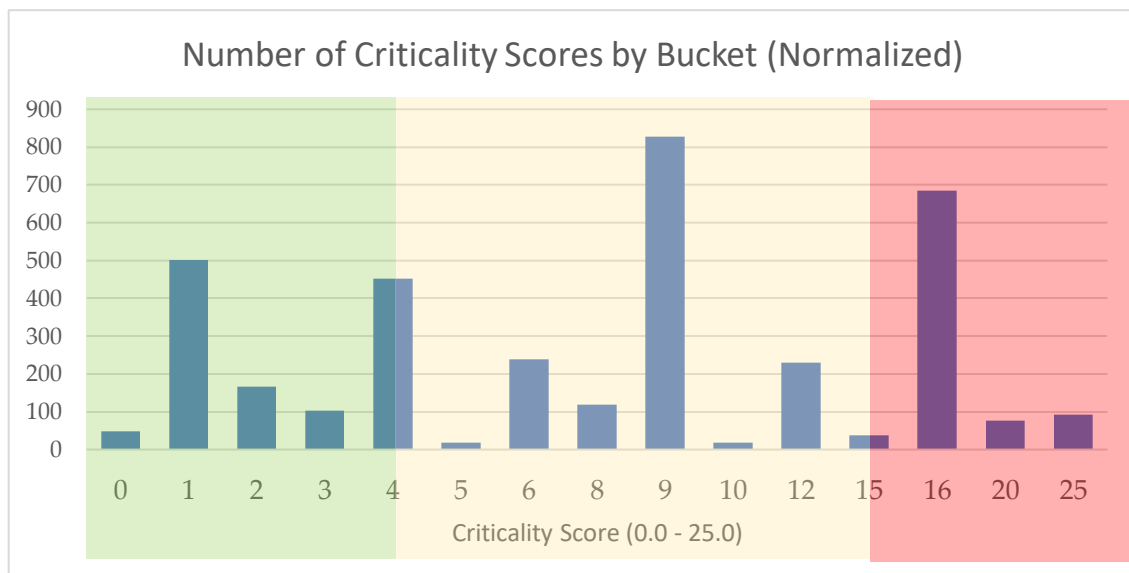


Figure 4: Criticality Scores by Bucket

To create the graph in Figure 4, the large dataset from Cobalt was analyzed and all criticality scores were calculated. Scores were normalized per the footnote below and graphed in R and

\* Note: As discussed in section 2.1.5, scores are calculated by multiplying an Impact score (0-5) by a Likelihood score (0-5), both integers. This may explain why there are no vulnerabilities with the score of 11 and other numbers seem more common. The graph normalizes for number of ways of a criticality score can be determined (ex. 9 can be 3x3, 5x4 and 4x5), but it cannot reflect the removal of scores like 11.

Excel. The graph indicates low, medium and high criticality using the green, yellow, red color scheme. The graph offers a number of takeaways regarding the perceived criticality of a large set of vulnerabilities, some of which are a function of Cobalt as a penetration testing platform. If a data set from the National Vulnerability Database was used instead, there would likely be a much higher percentage of vulnerabilities at the lower end of the criticality scores, due to the vast number of vulnerabilities publicly disclosed with no exploits. Since the goal of penetration testing is to provide valuable feedback for effective remediation, there is a surprising skew towards medium to high critical vulnerabilities. The entire dataset has an average criticality score of 7.16 out of 25, with 25% of vulnerabilities with a criticality score above 9, which is considered medium-high criticality.

A few other miscellaneous findings focused primarily on the remediation levels of the vulnerabilities found, or the perceived urgency and intention for fixing vulnerabilities. Of the entire dataset of over 6000 vulnerabilities, only 518 vulnerabilities, or less than 9%, had been labeled as "Not fix". This indicates that 91% of all vulnerabilities have been prioritized for fixes or have already been fixed, a whopping number considering that ~75% of the vulnerabilities are considered "medium" criticality or lower. Whether this indicates that Cobalt white-hat hackers are providing consistently strong findings or that companies err on the side of caution when it comes to security fixes, it certainly shows a commitment to fixing all vulnerabilities, even those that have low criticality scores. Later, industry averages for remediation are shown to be far lower than indicated on the Cobalt platform.

## 2. Vulnerability Scoring Systems

### 2.1 Existing Vulnerability Scoring Systems

#### 2.1.1 Why do we score vulnerabilities?

In the United States, debugging and patching security vulnerabilities costs companies over \$50 billion/year in lost productivity [11], exclusively for vulnerabilities with no attempted exploits. These patches may be tedious and expensive but patching a vulnerability that isn't attacked is far cheaper than the worst-case scenario, where the vulnerability is exploited in the wild. For reference, Facebook lost over \$13 billion in value *in one day*, after a data breach this past year [12]. While these breaches can be catastrophic, inefficient patching of vulnerabilities can be prohibitive for a company to grow.

It is incredibly difficult to patch all vulnerabilities in a system – no organization attempts to patch or fix every single vulnerability. In fact, less than 2% of all publicly published vulnerabilities have observed exploits in the wild [13]. However, if an organization is going to forgo patches on some vulnerabilities to mitigate losses in productivity, they need to know, with high certainty, which vulnerabilities are the most important to fix. Prioritizing vulnerabilities, or remediation, must be organized by some metric. This metric or score ties to a specific vulnerability, attempting to classify its importance in a set of scores.

Using the Exploit Prediction Model developed by Kenna Security, every decision to remediate a vulnerability is also a prediction about the future [13]. For every vulnerability that has not been exploited, the score associated with it, and consequently the prioritization of it in the remediation stack, is a prediction about the importance of that vulnerability. There are dozens of unique scoring systems, all with the same goal of creating a metric on which an organization can prioritize remediation efforts. Some scoring systems are necessary for compliance, such as Payment Card Industry Data Security Standards (PCI DSS), while others are for internal prioritization. This section will investigate a number of unique vulnerability scoring systems and recommend scoring systems that are most effective in remediation efforts.

#### 2.1.2 CVSS Base Score

A free and open vulnerability scoring system developed by the Forum of Incident and Response Teams (FIRST), the Common Vulnerability Scoring System, or CVSS, is largely considered the industry standard for determining severity scores for vulnerabilities [14]. The CVSS scoring system provides a score on a 0.0 to 10.0 scale, where 0.0 is the least severe vulnerability and 10.0 is the most severe vulnerability. The severity ranges fall as follows: None (0.0), Low (0.1 – 3.9), Medium (4.0 – 6.9), High (7.0 – 8.9) and Critical (9.0 – 10.0) [14].

These scores are developed through a number of different factors, inferring the likelihood of a vulnerability being exploited and the impact to the asset with the vulnerability if it were exploited. CVSS has gone through 3 primary iterations and it can be assumed that all references to CVSS in this thesis are CVSSv2.0, unless otherwise noted. CVSSv2.0 remains an accepted industry standard for the National Vulnerability Database and Open Source Vulnerability Database, among other security organizations. CVSSv3.0 follows similar logic as v2.0, with one major exception, replacing the Environmental Metric with the Modified Base Vector. Both v2.0 and v3.0 focus on different aspects of vulnerabilities, but for the sake of this thesis, v2.0 will be the focus.

CVSS assessment includes a Base Metric, a Temporal Metric and an Environmental Metric, all of which attempt to distinguish different vectors relevant in understanding the importance of a vulnerability. The CVSS Base Metric describes the vulnerability in a vacuum, essentially determining the intrinsic risk associated with this vulnerability type. The CVSS Base Metric returns a Base Score, developed from calculating the Exploitability and Impact of a vulnerability. These calculations are done as follows [15]:

$$(1) \textit{Exploitability} = 20 * \textit{AccessVector} * \textit{AccessComplexity} * \textit{Authentication}$$

$$(2) \textit{Impact} = 10.41 * (1 - (1 - \textit{ConfImpact}) * (1 - \textit{IntegImpact}) * (1 - \textit{AvailImpact}))$$

$$(3) f(\textit{impact}) = \begin{cases} 0 & \textit{Impact} = 0 \\ 1.176 & \textit{Impact} \neq 0 \end{cases}$$

$$(4) \textit{BaseScore} = \textit{roundToOneDecimal}(((0.6 * \textit{Impact}) + (0.4 * \textit{Exploitability}) - 1.5) * f(\textit{Impact}))$$

To calculate the values necessary in resulting in a Base Score, the following standards are used to identify the values of *Access Vector*, *Access Complexity*, and *Authentication* for Exploitability and *Confidentiality*, *Integrity*, and *Availability* for Impact [14, 15].

- *Access Vector* – Access Vector can take the value of 0.395, 0.646 or 1.0, if the access is Local, Adjacent Network, or Network, respectively. Descriptions of each of the three access vector types can be found in Appendix A.1.
- *Access Complexity* – Access Complexity can take the value of 0.35, 0.61, or 0.71, if the complexity is High, Medium, or Low, respectively. Descriptions of each of the three access complexity levels can be found in Appendix A.2.
- *Authentication* – Access Complexity can take the value of 0.45, 0.56, or 0.704, if the authentication system is Multiple Authentication, Single Authentication, or No Authentication, respectively. Descriptions of each of the three authentication levels can be found in Appendix A.3.

- *Confidentiality* – Confidentiality can take the value of 0.0, 0.275, or 0.660, if the attack has No Impact, Partial Impact, or Complete Impact, respectively, on the confidentiality of the system. Descriptions of each of the three confidentiality types can be found in Appendix A.4.
- *Integrity* – Integrity can take the value of 0.0, 0.275, or 0.660, if the attack has No Impact, Partial Impact, or Complete Impact, respectively, on the integrity of the system. Description of each of the three integrity types can be found in Appendix A.5.
- *Availability* - Availability can take the value of 0.0, 0.275, or 0.660, if the attack has No Impact, Partial Impact, or Complete Impact, respectively, on the integrity of the system. Descriptions of each of the three availability types can be found in Appendix A.6.

The Temporal Metric as a part of CVSS describes the risks associated with the vulnerability, directly associated with how the vulnerability will evolve over time [14, 15].

$$(5) \text{ TemporalScore} = \text{roundToOneDecimal} (\text{BaseScore} * \text{Exploitability} * \text{Remediation Level} * \text{ReportConfidence})$$

To calculate the values necessary in resulting in a *Temporal Score*, the following standards are used to identify the values of *Exploitability*, *Remediation Level* and *Report Confidence* [14, 15].

- *Exploitability* – Exploitability can take the value of 0.85, 0.9, 0.95, 1.0, or 1.0, if the attack has an Unproven, Proof-of-Concept, Functional, High or Undefined exploit technique, respectively
- *Remediation Level* – Remediation Level can take the value of 0.87, 0.90, 0.95, 1.0, or 1.0, if the attack has an Official Fix, a Temporary Fix, a Workaround, Unavailable Fix, or Undefined, respectively.
- *Report Confidence* – Report Confidence can take the value of 0.9, 0.95, 1.0, or 1.0, if the vulnerability is Unconfirmed, Uncorroborated, Conformed or Undefined, respectively.

The final metric is the CVSS Environmental Metric, which attempts to integrate risks associated with the vulnerability and how it is implemented or found in a specific asset or environment [14, 15].

$$(6) \text{ AdjustedImpact} = \text{MIN}(10, 10.41 * (1 - (1 - \text{ConfImpact} * \text{ConfReq}) * (1 - \text{IntegImpact} * \text{IntegReq}) * (1 - \text{AvailImpact} * \text{AvailReq})))$$

$$(7) \text{ AdjustedTemporal} = \text{TemporalScore},$$

*but recomputed with BaseScores Impact equation  
replaced with AdjustedImpact*



$$(8) \text{ EnvironmentalScore} = \text{roundToOneDecimal} ((\text{AdjustedTemporal} + (10 - \text{AdjustedTemporal}) * \text{CollateralDamagePotential}) * \text{TargetDistribution})$$

To calculate the values necessary in resulting in an *Environmental Score*, the following standards are used to identify the values of *Exploitability*, *Collateral Damage Potential* and *Target Distribution* [14, 15].

*Collateral Damage Potential* – Collateral Damage can take the value of 0.0, 0.1, 0.3, 0.4, 0.5, or 0, if the potential loss is described as None, Low, Low-Medium, Medium-High, High, or Not Defined, respectively.

*Target Distribution* - Target Distribution can take the value of 0.0, 0.25, 0.75, 1.0, or 1.0, if the proportion of systems potentially affected loss is described as None, Low, Medium, High, or Not Defined, respectively.

### 2.1.3 Limitations of CVSS

Despite the seemingly comprehensive CVSS scoring system, there are significant criticisms for this method of establishing a risk score for a vulnerability. CVSS is a generally satisfactory rating system in a vacuum, to score the severity of a Cross Site Scripting (XSS) vulnerability. However, most industry experts believe that the vectors intended to quantify statistical relevance for environmental and temporal impact are not representative and it is difficult to draw conclusions from them.

Joe Sechman, the VP of Penetration Test Delivery at Cobalt, cites concerns with attack chaining as the primary limitation of CVSS [32]. Attack chaining describes actions by a hacker or malicious actor to expose additional vulnerabilities through outcomes of a previous exploitation in the same system or other systems. For example, if a hacker is able to recover a list of passwords through a SQL Injection attack, this information can create an entire new set of vulnerabilities within that system and potentially new vulnerabilities for users in other systems. The CVSS calculations will certainly establish this vulnerability as severe through high Confidentiality, Collateral Damage Potential, and Target Distribution values. However, these stagnant values cannot fully describe the proliferation of new vulnerabilities due to this attack, as compared to another attack with the same score.

### 2.1.4 Bugcrowd Vulnerability Rating Taxonomy (VRT)

Bugcrowd is a crowdsourced cybersecurity platform focused on bug bounty programs [16]. In a bug bounty, an organization posts a program where hackers can submit vulnerabilities that they find and get paid by the hosting organization for finding them. To facilitate this process, Bugcrowd released the Vulnerability Rating Taxonomy (VRT) in 2016, specifically for its own cybersecurity platform. However, many industry experts, including

Mike Shema, the head of Product Security at Square, discuss VRT as one of the staples in bug bounty vulnerability scoring systems.

Less complex than CVSS Base Scoring, VRT is primarily a resource for bug hunters, focusing on vulnerabilities commonly found and accepted in bounty programs. VRT separates vulnerabilities into five prioritization categories from P1 – P5, which indicates the severity score of the vulnerability [16]. VRT clarifies that these scoring categories “do not equate to ‘industry accepted impact’”, but instead are intended to show a base severity rating for that vulnerability [16]. Mapping to the OWASP Top 10 (discussed in depth in 3.2), the VRT offers a clean and simple system to prioritize vulnerabilities based on what the weakness or exploit is. A simple 1 – 5 score, the Bugcrowd Vulnerability Rating Taxonomy is not a comprehensive scoring system, but it is an effective standard for the bug bounty industry.

### 2.1.5 Cobalt Labs

Cobalt is a crowdsourced penetration testing platform, offering penetration testing for organizations looking for vulnerabilities in their applications. Cobalt’s penetration testing system is built upon teams of 3-5 white-hat hackers across the globe that search for vulnerabilities and report them to companies who can remediate these vulnerabilities before they are exploited in production [17].

When reporting found vulnerabilities, Cobalt scores these vulnerabilities to provide detailed information on remediation priorities and levels of concern for any potential exploits. However, Cobalt uses a unique scoring system, related to CVSS but notably distinct, that offers an alternative to how we think about vulnerabilities and the scoring of them.

Cobalt white-hat hackers are asked to rate each vulnerability on two metrics: impact and likelihood. Impact describes the significance of an exploit of the vulnerability to the assets and the larger organization. Likelihood describes the probability that the vulnerability with both be found and exploited by a malicious actor. The scoring system requires a 0.0 – 5.0 score on each of these metrics, with 0.0 being low likelihood or impact and 5.0 being high likelihood or impact. The product of these two factors gives us a single criticality score for a given vulnerability:

$$(1) \text{ Criticality} = \text{Impact} * \text{Likelihood}$$

Ranging from 0.0 to 25.0, the criticality score attempts to convey the holistic significance of a found vulnerability by a Cobalt hacker. The two factors embedded in criticality, impact and likelihood, are commonly found in attempts to quantify the significance of a vulnerability. Far simpler than the CVSS Base Scoring system, Cobalt’s vulnerability scoring relies heavily on a hacker’s ability to properly assess the vulnerability from an objective perspective.

There are significant positives associated with this simplified approach to scoring vulnerabilities for a penetration testing platform. Unlike the bug bounty platforms like Bugcrowd or HackerOne, which often result in a large set of predominantly irrelevant or unimportant vulnerabilities, penetration testing tends to focus on fewer, but more significant vulnerabilities. For a bug bounty company like Bugcrowd, a system like VRT that identifies very specific vulnerabilities to report, or even a CVSS score, with a comprehensive set of inputs, is essential because most organizations will not remediate many of the reported vulnerabilities. With such a large set of vulnerabilities, a comprehensive system to prioritize vulnerability remediation is essential. However, for a company like Cobalt, or even a typical security consultancy, prioritizing remediations carries less weight.

This scoring system can be criticized for its lack of comprehensiveness in identifying vulnerability criticality. It offers enough information for an informed security expert to use this score as a base to determine whether remediation is necessary, and it avoids overcomplicating the score like some argue CVSS does. It may be sufficient for a penetration testing environment, but it should not be adopted as an industry standard or used to prioritize a large set of vulnerabilities.

### 2.1.6 Kenna Security

Kenna Security is a risk-based vulnerability management system, focusing on predicting future cyber-attacks and offering remediation recommendations to prevent them. Kenna's Exploit Prediction Model, referenced earlier, crystallizes the goal of Kenna's prioritization system. The model describes the decision to remediate a vulnerability as a prediction about the future; Kenna's classification system attempts to output predictions that can be used to prioritize vulnerability remediation [18]. Kenna's risk management approach is not described by a pure score attached to a vulnerability, but an application of other vulnerability scoring systems to a more comprehensive security strategy. CVSS scores capture the characteristics of a vulnerability, but Kenna answers if (and when) those characteristics warrant a patch.

This security strategy is measured through two essential metrics: **coverage** and **efficiency**. **Coverage** measures the completeness of a remediation process. Specifically, of all vulnerabilities that need to be fixed, what percentage of them were correctly identified and fixed? **Efficiency** measures the precision of a remediation process. Specifically, of all vulnerabilities that are identified to be fixed, what percentage were actually necessary to fix:

$$\text{Coverage \%} = \frac{(\text{Number of Vulns Fixed that Needed Fixes})}{(\text{Number of Vulns that Need Fixes})}$$

$$\text{Efficiency \%} = \frac{(\text{Number of Vulns Fixed that Needed Fixes})}{(\text{Number of Vulns Fixed})}$$

These two metrics will be used to evaluate remediation strategies through Kenna's platform.

Kenna's vulnerability remediation strategy, which functions as an augmented version of a base scoring system for vulnerabilities, is built upon a blend of those very base scoring systems. Conveniently described as the "Everything" model, the prioritization system was trained on (i) CVSS Base Vectors and computed scores, (ii) a list of Products and Vendor data, (iii) Category References on CVEs and (iv) keywords and phrases found in CVE descriptions<sup>†</sup>.

Kenna opens an interesting discussion, shifting the focus from theoretical vulnerability scores to the practical application of them. Instead of analyzing vulnerability scores alone, this thesis will now apply those scores to vulnerability remediation strategies.

## 2.2 Vulnerability Remediation Strategies

The goal of a vulnerability remediation strategy (and the ultimate goal of a vulnerability scoring system) is to fix vulnerabilities in "a cost-effective manner before they lead to security incidents" [18]. To best focus remediation efforts, it is essential to understand the stages in the *lifecycle* of a vulnerability. With the perspective of the entire vulnerability lifecycle, it is easier to recognize trends in exploits and recommend optimal remediation strategies. Using Kenna's report titled *Prioritization to Prediction: Volume 1* as a guide, we can follow the vulnerability lifecycle through roughly six stages [18]:

- *Vulnerability Created* – When flawed code is written, or when defective code is released into a system, a vulnerability has been created and exists. Vulnerabilities can and often do exist without ever being discovered or exploited.
- *Vulnerability Discovered* – This stage is fairly obvious. When a human or a scanner finds a vulnerability in a system, it is, by definition, discovered.
- *Vulnerability Disclosure* – After a vulnerability is discovered, it is often reported. Occasionally these vulnerabilities will be exclusively reported within an organization, if it is found by an employee or a proprietary penetration test. However, in many cases, vulnerabilities are disclosed publicly and are reported to the CVE or another public database. Once a vulnerability has been disclosed publicly, it is far more likely to be exploited, but only 33% of all vulnerabilities in CVE have been observed in live enterprise environments.
- *Vulnerability Exploit Code* – After a disclosure, there may be a working code exploit or framework to exploit the published vulnerability. These exploits can be posted publicly or developed privately, but only 2% of all published vulnerabilities have observed exploits in the wild.
- *Vulnerability Exploitation* – When a vulnerability is exploited in the wild, the impact of that exploit will vary based on the environment and vulnerability management systems in place to limit the impact. Similar vulnerabilities are rarely exploited simultaneously, so there is significant value in analyzing exploits of known vulnerabilities elsewhere to your organization's own system.

---

<sup>†</sup> Each of these four remediation strategies are described in Section 2.2 below in greater detail.

- *Vulnerability Detection Signature* – If a vulnerability is exploited in your system, it is absolutely essential to recognize this failure quickly. However, most detection systems need guidance for what to look for. If your remediation strategies are not effective, this late stage in the vulnerability lifecycle is the final line of defense against compounding issues from an exploit.

Not all vulnerabilities necessarily follow this lifecycle, but it offers a strong framework to address remediation strategies. Prioritizing which vulnerabilities to fix is a task that starts before vulnerabilities are even created and needs to continue throughout the existence of the system. The following remediation strategies are analyzed on the metrics of *efficiency* and *coverage* using the CVE as a database of vulnerabilities. These results are using the model developed by Kenna Security and indicate average successes of each remediation method [18].

### 2.2.2 CVSS Base Score

This is the most intuitive remediation strategy, as it takes a CVSS score and assumes the risk of the less significant vulnerabilities, while focusing on fixing the most impactful vulnerabilities. In fact, this remediation strategy is used by PCI, as discussed earlier. To reach full compliance, PCI requires that all related vulnerabilities with a CVSS score above 4 be remediated [31]. Deduced from Figure 5, this is an incredibly stringent and labor-intensive remediation strategy that will offer impressive coverage of vulnerabilities. Since PCI-compliant organizations deal with highly sensitive data and need compliance to operate, this high coverage requirement makes sense. However, this incurs a number of unnecessary costs and takes significant time to complete, with an efficiency level likely below 26%.

	Remediated correctly (True Pos.)	Delayed incorrectly (False Neg.)	Remediated too soon (False Pos.)	Delayed correctly (True Neg.)	Efficiency (Precision)	Coverage (Recall)	Efficiency by Chance	Coverage by Chance
10	1,510	20,207	5,025	67,855	<b>23.1%</b>	<b>7%</b>	23%	7.1%
9	3,148	18,569	10,405	62,475	<b>23.2%</b>	<b>14.5%</b>	23%	14.7%
8	3,228	18,489	10,736	62,144	<b>23.1%</b>	<b>14.9%</b>	23%	15.1%
7	11,562	10,155	25,180	47,700	<b>31.5%</b>	<b>53.2%</b>	23%	39.8%
6	14,320	7,397	34,715	38,165	<b>29.2%</b>	<b>65.9%</b>	23%	53.2%
5	17,547	4,170	49,753	23,127	<b>26.1%</b>	<b>80.8%</b>	23%	73%

Source: Kenna / Cyentia

Figure 5: Prioritization strategies by CVSS Base Score [18]

While a PCI-compliant strategy towards remediation seems comforting due to its coverage of nearly all vulnerabilities, it is simply unrealistic to expect all organizations to remediate at an ~80% coverage level. Moving up the table to lower coverage remediation strategies, a clear drop is seen, which is simply the percentage of CVEs that are found at each score level. The efficiency levels rise and actually peak at remediation at CVSS 7.0+ and then begin to drop again. Based on these two metrics, the optimal CVSS Base Score remediation strategy is to remediate all vulnerabilities with a CVSS score of 7.0 or higher.

### 2.2.3 Product and Vendor Data

An alternative vulnerability remediation strategy is to fix vulnerabilities from specific vendors and product offerings. While CVSS remediation strategies attempt to take a risk score and use it to prioritize remediation based on risk, this strategy is ostensibly less thorough. A small set of vendors does make up a disproportionately large set of CVEs, particularly companies like Microsoft, Oracle, Adobe and IBM. These organizations also provide some of the most commonly used, exploited and patched programs, which may be the primary vulnerability points for organizations using these products [18].

	Remediated correctly (True Pos.)	Delayed incorrectly (False Neg.)	Remediated too soon (False Pos.)	Delayed correctly (True Neg.)	Efficiency (Precision)	Coverage (Recall)	Efficiency by Chance	Coverage by Chance
Remediate Vendors Top5	2,598	19,119	18,500	54,380	<b>12.3%</b>	<b>12%</b>	23%	22.9%
Top10	3,588	18,129	27,705	45,175	<b>11.5%</b>	<b>16.5%</b>	23%	33.9%
Top20	4,726	16,991	34,471	38,409	<b>12.1%</b>	<b>21.8%</b>	23%	42.5%

Source: Kenna / Cyentia

Figure 6: Prioritization strategies by top vendors [18]

Seen in Figure 6, no strategy that focuses on top vendors or products is particularly efficient or encompassing. There are companies that will focus their remediation efforts on simply staying up to date on patches with these major vendors. It is extremely ineffective and not even efficient to do this. Even if we take the top 20 vendors, which will likely include a large portion of systems in most companies, only 21.8% coverage is met. For any organization, this should be inexcusably low coverage. Remediation by top vendors and products may not be effective, but we will later look at its positive impact on coverage and efficiency in a multi-faceted remediation system.

## 2.2.4 Category Reference Lists (on CVE ID)

The third remediation strategy is directly tied to the CVE database (see section 3.2.2 for more details on CVE), specifically the list of references cited on each ID. These references can contain all kinds of information, often related to vendor acknowledgement, additional published information, the initial disclosure of the vulnerability or a number of other things [18]. This strategy is rarely used alone, but it offers valuable insight into which CVEs are commonly tagged or recognized outside the CVE database. Commonly found tags are *sectrack* (linked to <https://securitytracker.com>), *ms* (linked to Microsoft reference), *fulldisc* (linked to Full-Disclosure mailing list), and *bid* (linked to Security Focus bugtraq ID), among others.

	Remediated correctly (True Pos.)	Delayed incorrectly (False Neg.)	Remediated too soon (False Pos.)	Delayed correctly (True Neg.)	Efficiency (Precision)	Coverage (Recall)	Efficiency by Chance	Coverage by Chance
sectrack	3,976	17,741	20,688	52,192	<b>16.1%</b>	<b>18.3%</b>	23%	26.7%
ms	1,054	20,663	2,714	70,166	<b>28%</b>	<b>4.9%</b>	23%	4.1%
fulldisc	1,300	20,417	2,308	70,572	<b>36%</b>	<b>6%</b>	23%	3.9%
confirm	5,296	16,421	43,480	29,400	<b>10.9%</b>	<b>24.4%</b>	23%	52.9%
bugtraq	7,695	14,022	12,139	60,741	<b>38.8%</b>	<b>35.4%</b>	23%	21.5%
bid	16,100	5,617	39,462	33,418	<b>29%</b>	<b>74.1%</b>	23%	60.2%

Source: Kenna / Cyentia

Figure 7: Prioritization strategies by reference lists [18]

Based on the efficiency and coverage analysis done by Kenna, most of these strategies do not offer above average coverage and efficiency balances, but are significantly better than random remediation, which means that they could provide positive data for a blended remediation method that includes these reference lists as additional data for prioritization.

## 2.2.5 Keywords and Phrases (on CVE Descriptions)

This is not a bona-fide remediation strategy on its own, but much like the reference lists, provides information that could be productive in a remediation strategy. All CVEs have descriptions, some longer than others, written in free form text. Using keywords and phrases from this free text may add additional information to a prioritization schedule [18]. The Kenna Security model includes these keywords in their strategy, but it is unclear if this remediation strategy is worth pursuing as anything more than a supplement.

## 2.3 Employee Entity Scoring Systems

Briefly shifting focus away from typical security vulnerabilities, it is important to recognize non-technical vulnerabilities within an organization. In a paper titled *Human*

*Capability Evaluation Approach for Cyber Security in Critical Industrial Infrastructure*, Ani, et al. discuss the importance of viewing employees as possible vulnerabilities themselves [19]. According to a research report published by PricewaterhouseCoopers, 36% of all malicious cyberattacks in 2013 were caused by human errors [19]. Human entities are the most targeted asset in cyberattacks by a significant margin.

According to the Verizon's Data Breach Report from 2018, there were over 43,000 accesses using stolen credentials through phishing in the past year [20]. The importance of security "awareness" has only grown since these statistics have been published. However, notable data breaches and exploits continue to occur, despite increasing awareness of cybersecurity threats. As discussed by Ani, et al., security "awareness" is meant to describe thoughtfulness on security, which encourages employees to flag security concerns and respond to them [19]. Increasing awareness of employees, particularly ones with elevated levels of access to assets, is essential to improving system security. This section will summarize attempts to quantify security awareness of human assets within an organization.

Labeled as Workforce Cybersecurity Capability (WCSC), the evaluation model for human entity cyber awareness is a function of an employee's knowledge and skill. Specifically, knowledge is "*the measure of information and theoretical understanding about...vulnerabilities...that a user, employee or operator is working with.*" Skill is "*the ability to...spot or detect cyber-attack attempts, patterns and techniques...and the degree, in which the user can respond timely with appropriate countermeasures*" [19]. Using both of these metrics, we can begin to estimate the cyber security readiness of an organization and its employees.

To collect data on an employee, Ani, et al. recommends questionnaires, interviews, general observations, attack simulations or penetration testing and gamification [19]. Of these strategies, penetration testing is the most comprehensive and indicative of real-world actions by employees, but also incurs the largest cost. A combination of these data collection methods is recommended, but none are particularly quantitative enough to develop a relative scoring system for the WCSC.

From here, the quantitative ranking of human security preparedness traits become less mathematically relevant and would likely be ineffective in relative scoring of employee security awareness. However, the concepts of *knowledge* and *skill*, with respect to the WCSC, are important enough to be included in all employee onboarding processes. Similarly, discussing security awareness as an organizational priority is essential to executing security procedures. While the research on scoring systems for employee security awareness seems unproven, the concepts discussed are necessary in understanding the human element in successful exploits and breaches.



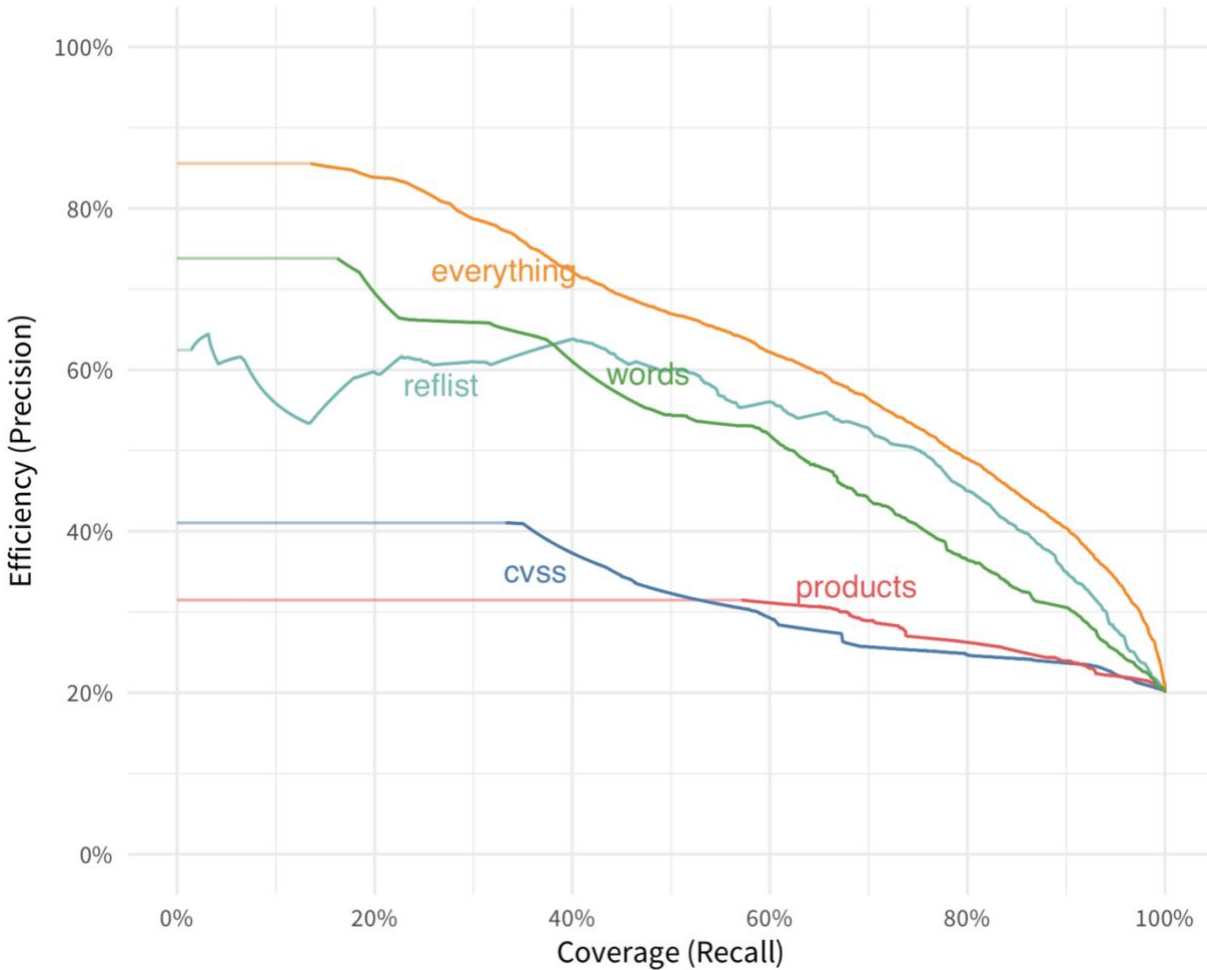
## 2.4 Conclusions: Vulnerability Scoring and Prioritization Methods

The ultimate security goal of any organization is to minimize total impact from security exploits. Organizations fix vulnerabilities to reach this goal. However, the key is ensuring that the correct vulnerabilities are fixed to minimize impact of those exploits. To maximize the likelihood of fixing the correct vulnerabilities, organizations prioritize remediation efforts by scoring vulnerabilities relative to one another. At the most granular level, the focus of this section is vulnerability scoring systems, the crux of minimizing security risk in an organization. Simply put, organizations cannot from cybersecurity attacks without robust vulnerability scoring systems to remediate most effectively.

Different organizations will have different standards and requirements for remediation, but any systems with the potential to become vulnerable in the future must have a strategy. Thus, it is difficult to offer any industry standard vulnerability remediation system. Similarly, it is difficult to generalize any vulnerability scoring system. However, the Kenna Security model provides metrics to quantitatively review remediation efforts and compare those efforts to other strategies.

Taking a number of the remediation strategies discussed, Kenna developed an “Everything” model, tested on CVSS base scoring, reference lists, vendors and products, and keywords found in CVE descriptions. Each of these factors provides additional information about past “performance” of vulnerabilities and can help maximize likelihood of effective remediation based on it. Seen in the graph in Figure 8, this model offers a more robust remediation strategy with respect to efficiency - coverage tradeoffs. If an organization is looking for the most comprehensive remediation strategy, for all coverage levels up to 100%, the “everything” model offers a more efficient route. Conversely, for all relevant efficiency levels up to ~85%, the “everything” model maximizes the coverage of it. Organizations will have internal valuations of coverage and efficiency, but the “everything” model is an efficient remediation frontier for all of them.

The model developed by Kenna is not the end-all-be-all of vulnerability remediation, but there are a few notable takeaways from this graphical representation of coverage and efficiency. Most interestingly, the CVSS scoring model is one of the least effective strategies, despite being one of the most detailed and robust scoring systems. Noting this, is it fair to question whether CVSS is properly estimating risk in its scoring system? Or does CVSS scoring and subsequent prioritization limit the ability to efficiently remediate vulnerabilities? These are questions that require a bit more investigation but will be discussed briefly in section 5.



Source: Kenna / Cyentia

Figure 8: Coverage/Efficiency tradeoffs for remediation models [17]

Concluding this section on scoring systems and remediation strategies, it is important to keep a balance between academic value and practical implementation. Academic approaches to scoring systems, like CVSS, can attempt to quantify risk of a vulnerability in a singular number, but oftentimes are not aligned with organizational security needs. A base scoring system is inherently difficult to tailor to specific business needs. Conversely, Bugcrowd’s VRT focuses on a simple 1-5 categorization for bug bounties. This is an effective (and widely accepted) practical use, but it does not comprehensively define the security impact of each vulnerability. “Everything” prioritization systems, like the one developed by Kenna, can fit business efficiency requirements, while maximizing security coverage.

## 3. Vulnerability Classifications and Taxonomies

### 3.1 Vulnerabilities, Exploits, Threats and Taxonomy Types

#### 3.1.1 Why Categorize Vulnerabilities, Threats and Exploits?

After a weakness is found in a system, the first step to eliminate that weakness is understanding what the weakness is. Classification systems, such as the Common Weakness Enumeration (CWE), offers a reference guide to security researchers and software engineers focused on patching these vulnerabilities. Categorization of vulnerabilities and exploits can simultaneously facilitate security awareness within an organization and functionally support the CISO's efforts in minimizing total impact of security flaws.

However, categorization is a very general term. Looking at large datasets of vulnerabilities, such as the one analyzed from Cobalt in Section 1, there are plenty of effective ways to separate security concerns into effective categories. There are **defense-focused systems** like the SANS Top 20. There are massive **vulnerability taxonomies** like the CVE and NVD. There are **weakness enumerations** like the CWE, which focuses more on classification. There are **threat taxonomies**, focused on the types of approaches that deliver or execute exploits. And there are representations of all these with **bucketing systems** like the OWASP Top 10.

Regardless of the way vulnerabilities and weaknesses are categorized, the goal of categorization is to emphasize specific aspects about the vulnerabilities, in turn providing better understanding for how to defend against them. This section of the thesis focuses on the pros and cons of each type of vulnerability/weakness/threat/exploit categorization and reflects on the most effective ones.

#### 3.1.2 Vulnerabilities vs. Exploits

Before delving into specific classification standards, a distinction needs to be made between a vulnerability and an exploit, as classification systems will deal with each of these entities differently. This can be most easily described by an example from the Common Weakness Enumeration (CWE is discussed in more detail in section 3.2.1).

Most vulnerabilities in the CWE have no associated or published exploit, so let us look at CWE-611: Improper Restriction of XML External Entity Reference, or XXE [21]. This vulnerability takes advantage of XML entity definitions, where an attacker can define a file path that will lead an application to read the contents of a local file, potentially maliciously. The vulnerable application will process the XML and may expose the contents of a file that may be outside its sphere of control [21].

Here, the vulnerability is clear. The processing of the XML document contains the vulnerability and impact of this weakness may be significant, with detrimental effects to Confidentiality, Integrity and Availability of the system. The exploit may be slightly less clear. As described on the CWE page, a URI in an XML file such as `file:///etc/passwd` designates the password file in Unix-based systems. This is an example of an exploit. Using this example, it should be evident that a vulnerability is a flaw in the system, regardless of its inception. Vulnerability is generally used interchangeable with weakness, but in the case of CVE and CWE, distinctions emerge. An exploit or attack is the act of attempting to take advantage of the specific vulnerability and can be done in this example by providing a malicious XML file.

A vulnerability (or weakness) is the poorly guarded door and an exploit is the strategy or path taken to gain access to that door to take advantage of its weakness. Some taxonomies and categorization systems will focus on the vulnerability or weakness, while others will focus on the exploit or threat. The most important takeaway is that a singular published exploit can jeopardize the security of thousands of systems, through just one specific vulnerability.

### 3.1.3 Threat Taxonomies

Threat taxonomies focus on the act that delivers or propagates an exploit. This is distinctly different than defense-focused taxonomies (SANS Top 20), vulnerability enumerations (CWE), and vulnerability classification buckets (OWASP Top 10). The primary threat taxonomy is developed by ENISA, the European Union Agency for Network and Information Security [22]. With the goal of classifying threats on their type, it focuses on the top threats from over 100 classes of threats, based on the source of security issues and the environments in which the issues appear.

The ENISA threat taxonomy does not consider the *type* of attack or whether the malicious threat is delivered as an XSS attack, but instead considers *how* that threat manifests. For example, this would focus on the payload delivering the XSS attack, if it is a malicious phishing email or downloaded malware. ENISA focuses on a top 15 set of threats each year, with their Top 15 Threats from 2017 – 2018 detailed below [22].

Unsurprisingly, the most common threats faced remain fairly consistent, while the development of new vulnerability exploitation methods emerge in the forms of botnets and cryptojacking. Similar to the SANS Top 20, the ENISA threat taxonomy provides essential support to organizations attempting to improve employee security awareness. Most of these threats to systems target human entities as the initial contact for the exploit, so understanding these avenues is essential to improving organizational security.



















































Top Threats 2017	Assessed Trends 2017	Top Threats 2018	Assessed Trends 2018	Change in ranking
1. Malware		1. Malware		
2. Web Based Attacks		2. Web Based Attacks		
3. Web Application Attacks		3. Web Application Attacks		
4. Phishing		4. Phishing		
5. Spam		5. Denial of Service		
6. Denial of Service		6. Spam		
7. Ransomware		7. Botnets		
8. Botnets		8. Data Breaches		
9. Insider threat		9. Insider Threat		
10. Physical manipulation/ damage/ theft/loss		10. Physical manipulation/ damage/ theft/loss		
11. Data Breaches		11. Information Leakage		
12. Identity Theft		12. Identity Theft		
13. Information Leakage		13. Cryptojacking		<b>NEW</b>
14. Exploit Kits		14. Ransomware		
15. Cyber Espionage		15. Cyber Espionage		
<b>Legend:</b> Trends:  Declining,  Stable,  Increasing Ranking:  Going up,  Same,  Going down				

Figure 9: ENISA Threat landscape (2017, 2018) [22]

Focusing less on the specific delivery mechanisms (phishing, malware, etc.) for exploits, another emphasis for threat taxonomies is the asset under scrutiny. No systems in any organization are exclusively code, they are often developed in a complex environment with physical assets, hardware, human interaction and a number of other subsections that can be the primary target for an exploit. This simplified taxonomy is shown below to describe types of threats, when simplified and organized by asset [23]:

- Physical Security – Assets come in contact with a threat actor, potentially losing integrity, availability or confidentiality of the system.
- Hardware Security – A potential loss of availability, hardware failures can focus on hard disk drives, CPUs, routers, network printers, or other connected devices.

- Software Security – The most typical threat, where issues originate from code defects or faulty code, making the system behave in a way that the organization did not intend.
- Crypto and Protocol Security – Failures can result in loss of confidentiality of the system due to insecure use of cryptographic systems to protect data in rest, transit and use.
- Interoperability – Software updates that contain changes can break interoperability of systems, which can cause loss of availability. When assets interact with one another in the system, interoperability is essential to intra-system communication.
- Configuration Security – One of the vaguer threats modeled, this describes issues associated with settings or prescribed behaviors of a system that leave it open to exploits.
- Supply Chain Security – The use of vendor provided software, third party packages and other hardware installations force organizations to rely on their supply chain partners' security to maintain their own security.
- Human Error – As described numerous times before, human actions within a complex system can result in negative outcomes that are not intended by anyone. This focuses less on a malicious insider and more on human misuse or low security awareness causing errors.

## 3.2 Classification Standards & Sources

### 3.2.1 Common Weakness Enumeration (CWE)

Developed and maintained by MITRE, the Common Weakness Enumeration (CWE) is “a community-developed list of common software security weaknesses” [24]. With the goal of being a baseline for weakness identification, CWE offers a comprehensive list of all publicly disclosed software weaknesses. Each entry describes a specific type of weakness and is assigned a CWE ID number. Each of these CWE IDs identifies a specific type of weakness with a detailed description, membership classes, buckets and any known exploits or patches publicly disclosed.

Each CWE ID offers the scope, potential impact, membership groupings and other valuable information for the associated weakness. The goal of the CWE is to identify and understand vulnerabilities and software flaws as they are found, to improve likelihood of preventing these flaws. The CWE has over 600 unique IDs describing software weaknesses and vulnerabilities, but the enumeration itself has over 1000 unique IDs describing things beyond just weaknesses. For example, CWE-710 is described as Improper Adherence to Coding Standards, an umbrella CWE with a number of specific weaknesses encompassed underneath it. Combined with the specific weaknesses such as CWE-45, these category and bucket CWE IDs make up the entirety of the enumeration.

## CWE-45: Path Equivalence: 'file...name' (Multiple Internal Dot)

Weakness ID: 45

Abstraction: Variant

Structure: Simple

Status: Incomplete

Presentation Filter:

### ▼ Description

A software system that accepts path input in the form of multiple internal dot ('file...dir') without appropriate validation can lead to ambiguous path resolution and allow an attacker to traverse the file system to unintended locations or access arbitrary files.

Figure 10: CWE-ID 45 from <https://cwe.mitre.org/data/definitions/45.html> [24]

The direct benefits of the Common Weakness Enumeration are significant. Most notable and most relevant for this section is the common language it has developed for “discussing, finding, and dealing with the causes of software security weaknesses as they are manifested in code, design or architecture” [24]. In addition, the public display of these weaknesses allows for organizations to offer clear and organized claims of security weaknesses that may exist in products that they develop or curate. Not exclusively a taxonomy, CWE also offers code examples and other instructional tools that can help educate users on the types of weaknesses and how they may appear in code.

The CWE offers tremendous value as vulnerability taxonomy and categorization system. The CWE functions as a classification tool that should be able to properly describe any software vulnerability found in a system or application. Having a one-to-one relationship from CWE to security issue allows for bucketing of weaknesses and well-organized list of all known weaknesses in a singular mapping. A weakness in a system that is found and cannot be mapped to an existing CWE ID is fairly rare. To show the comprehensiveness of the CWE classification system, this thesis developed a makeshift mapping from 2017 OWASP Top 10 Buckets to the CWE. This can be seen in Figure 11, with a sample set of the most common CWEs and how the CWE taxonomy is a wholly comprehensive one.

OWASP Top Ten (2017)										
A1-Injection	A2-Broken Authentication	A3-Sensitive Data Exposure	A4-XML External Entities	A5-Broken Access Control	A6-Security Misconfiguration	A7-Cross Site Scripting	A8-Insecure Deserialization	A9-Components with Known Vulnerabilities	A10-Insufficient Logging & Monitoring	Not Directly Mapped to OWASP Top 10
<b>CWE --&gt; OWASP Top Ten (2017) Mapping</b>										
Command Injection	Improper Authentication	Sensitive Data Under FTP Root	XXE	Path Traversal	Information Exposure - Error Messages	Cross Site Scripting	Deserialization of Untrusted Data		Omission of Security-relevant information	C8-Other
OS Command Injection	Unprotected Storage of Credentials	Improper Certificate Validation	XML Entity Expansion	Improper Access Control	Information Exposure - Directory Listing			Insufficient Logging		C10-Remote Code Execution
Argument Injection Modification	Single Factor Authentication	Missing Encryption of Sensitive Data		Improper Authorization						C9-Redirects and Forwards
LDAP Injection	Session Fixation	Cleartext storage of sensitive info.		Direct Request						?
XML Injection	Insufficiently Protected Credentials	Cleartext transmission of sensitive info.		Cleartext transmission of sensitive info.						
Expression Language Injection	Unprotected Transport of Credentials	Missing Required Cryptographic Step								
Data Query Injection	Insufficient Session Expiration	Inadequate Encryption Strength								
	Unverified Password Change	Use of Broken Cryptographic Algorithm								
	Weak Password Recovery Mechanism	Reversible One-Way Hash								
		Exposure of Private Information								

Figure 11: OWASP Top 10 to Cobalt Buckets to CWE Mapping

Referring back to the dataset analysis in the first section, of the 450 vulnerabilities from Cobalt’s platform that were not categorized into the modified-OWASP Top 10, all 450 were mapped properly to specific CWE IDs. The comprehensiveness of the CWE is particularly valuable for any vulnerability classification system and there are no weakness-specific taxonomies that can as completely cover found vulnerabilities. The CWE includes a number of scoring systems in its own right, but these systems, such as CWSS, offer very little distinction from the scoring systems and remediation processes already discussed.

### 3.2.2 Common Vulnerabilities and Exposures (CVE)

The Common Vulnerabilities and Exposures (CVE) is a list of all publicly known cybersecurity vulnerabilities [25]. Also developed and maintained by MITRE, the CVE is often discussed in tandem with the CWE, but the distinction between the two entities is important. CWE focuses on vulnerabilities, or the underlying flaw in code or within a system. CVE focuses on vulnerabilities and the actual exposures of them -- the specific instances of the vulnerability that are found within a product or asset. With over 115,000 entries in the CVE, it is a large and diverse list of publicly disclosed vulnerabilities [25]. Each CVE can be directly mapped to an associated CWE ID that will describe the weakness associated with the real-world exposure. Each CVE entity is similarly described, with a description of the vulnerability, references to the disclosure and any additional information attached to that specific exposure. An example of a CVE entry from this past year is shown in Figure 12.



CVE-ID	
<b>CVE-2019-1721</b>	<a href="#">Learn more at National Vulnerability Database (NVD)</a> • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
<p>A vulnerability in the phone book feature of Cisco Expressway Series and Cisco TelePresence Video Communication Server (VCS) could allow an authenticated, remote attacker to cause the CPU to increase to 100% utilization, causing a denial of service (DoS) condition on an affected system. The vulnerability is due to improper handling of the XML input. An attacker could exploit this vulnerability by sending a Session Initiation Protocol (SIP) message with a crafted XML payload to an affected device. A successful exploit could allow the attacker to exhaust CPU resources, resulting in a DoS condition. Manual intervention may be required to recover the device. This vulnerability is fixed in Cisco Expressway Series and Cisco TelePresence Video Communication Server Releases X12.5.1 and later.</p>	
References	
<p><b>Note:</b> <a href="#">References</a> are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.</p>	
<ul style="list-style-type: none"> <li>• <a href="#">BID:108016</a></li> <li>• <a href="http://www.securityfocus.com/bid/108016">URL:http://www.securityfocus.com/bid/108016</a></li> <li>• <a href="#">CISCO:20190417 Cisco Expressway Series and Cisco TelePresence Video Communication Server Denial of Service Vulnerability</a></li> <li>• <a href="https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20190417-es-tvcs-dos">URL:https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20190417-es-tvcs-dos</a></li> </ul>	

Figure 12: CVE2019-1721 from <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-1721> [25]

CVE is an incredibly useful and widely applied tool, focused less on categorization of vulnerabilities and more on identification of them. The CVE IDs are commonly used to identify a vulnerability found within a system and is used in the remediation model described by Kenna Security in section 2.

CVEs can be identified by MITRE themselves, with MITRE functioning as an editor of sorts for the list of all exposures, but the most notable value comes from organizations that assign CVE IDs for their own products. Large entities in the software space, particularly companies like Microsoft, IBM, Oracle, etc.) will designate CVEs for flaws identified and remediated within their own products. This may not be an immediate posting, but the ability to maintain your own security by following CVE updates for products within your organization is incredibly valuable. Since all CVEs are descriptors of weaknesses in products that have already been released, any organization that uses external software or products relies on CVE as a tool for staying current on weaknesses that may exist in their own systems.

To develop a bit more context for the relationship between CVEs and CWEs, see Figure 13. On the left, the top products found in the CVE are shown, some of them self-reported by large companies and public entities [26]. This reinforces the association between a large number of vulnerabilities in the CVE with single products or systems. In the table on the right, there is an association between CWE IDs and the number of CVEs found for each one. This indicates the one-to-many relationship previously described between the CWE and the CVE. Buffer Overflow, one of the older and less complex weaknesses in the enumeration, still has manifested in over 530 vulnerabilities publicly disclosed in the CVE [26]. These tables offer

important context on these CVE/CWE relationships and valuable insight into which types of weaknesses and products are most vulnerable. Tying another industry standard into the CVE, the next section will focus on the National Vulnerability Database.

	Top Products	Num. CVEs
1.	Linux Kernel	917
2.	Ubuntu	211
3.	FFmpeg	187
4.	Debian	170
5.	Wireshark	146
6.	openSUSE	134
7.	PHP	125
8.	Android	121
9.	Fedora	105
10.	QEMU	77

	CWE ID	Weakness Summary	Num. CVEs
1.	119	Buffer Overflow	539
2.	20	Improper Input Validation	390
3.	264	Access Control Error	318
4.	79	Cross-Site Scripting	273
5.	200	Information Disclosure	228
6.	189	Numeric Error	221
7.	399	Resource Management Error	219
8.	362	Race Condition	72
9.	89	SQL Injection	61
10.	310	Cryptographic Issues	42

Figure 13: Left: Top software products by number of CVE IDs. Right: Top CWE weaknesses by number of CVE IDs [26]

### 3.2.3 National Vulnerability Database (NVD)

The National Vulnerability Database (NVD) is a repository of standards-based vulnerability management data, curated and maintained by the US government. Specifically, the NVD is curated by NIST, the National Institute of Standards and Technology, providing a particularly valuable status as a database of record [27]. Linked to both the CVE and the CVSS, NVD offers detailed information regarding found vulnerabilities from the CVE.

One area where the NVD differs from most from the other databases and classification organizations is in its focus on industry verticals. As discussed in an interview with Joe Sechman (see section 4), he indicated significant value in the focus on Healthcare and Financial Sector vulnerabilities within the NVD and how these vertical identifications can improve remediation strategies. For example, a simple search in the NVD for “Healthcare” offers 62 relevant vulnerabilities, all looking at healthcare tools or products that contain publicly disclosed vulnerabilities. All of this information stored in a singular place functions as a source of record for public vulnerabilities.

Each NVD entry is labeled with an associated CVE ID, description of the vulnerability and a vast amount of information associated with it. Most notably, we find the Impact scores of the vulnerability, CVSS v2.0 and CVSS v3.0. Section 2.1.2 of this thesis focuses more heavily on CVSS v2.0, but as seen in the example NVD entry in Figure 14, both of the scores are included here because a number of organizations still tend to use v2.0. Beneath the Impact scores and the description, the NVD may reference to potential mitigants of the vulnerability, particularly when it is a self-reported or diagnosed vulnerability by an organization themselves (Cisco in this case). Less of a taxonomy than the other strategies discussed in this section, NVD still offers incredible value as a resource for identifying and describing vulnerabilities in a searchable manner.

## CVE-2019-1721 Detail

### Current Description

A vulnerability in the phone book feature of Cisco Expressway Series and Cisco TelePresence Video Communication Server (VCS) could allow an authenticated, remote attacker to cause the CPU to increase to 100% utilization, causing a denial of service (DoS) condition on an affected system. The vulnerability is due to improper handling of the XML input. An attacker could exploit this vulnerability by sending a Session Initiation Protocol (SIP) message with a crafted XML payload to an affected device. A successful exploit could allow the attacker to exhaust CPU resources, resulting in a DoS condition. Manual intervention may be required to recover the device. This vulnerability is fixed in Cisco Expressway Series and Cisco TelePresence Video Communication Server Releases X12.5.1 and later.

Source: MITRE

Description Last Modified: 04/17/2019

[View Analysis Description](#)

### QUICK INFO

CVE Dictionary Entry:

CVE-2019-1721

NVD Published Date:

04/17/2019

NVD Last Modified:

04/23/2019

### Impact

#### CVSS v3.0 Severity and Metrics:

Base Score: 6.5 MEDIUM

Vector: AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H (V3 legend)

Impact Score: 3.6

Exploitability Score: 2.8

Attack Vector (AV): Network

Attack Complexity (AC): Low

Privileges Required (PR): Low

User Interaction (UI): None

Scope (S): Unchanged

Confidentiality (C): None

Integrity (I): None

Availability (A): High

#### CVSS v2.0 Severity and Metrics:

Base Score: 6.8 MEDIUM

Vector: (AV:N/AC:L/Au:S/C:N/I:N/A:C) (V2 legend)

Impact Subscore: 6.9

Exploitability Subscore: 8.0

Access Vector (AV): Network

Access Complexity (AC): Low

Authentication (AU): Single

Confidentiality (C): None

Integrity (I): None

Availability (A): Complete

Additional Information:

Allows disruption of service

Figure 14: NVD CVE-2019-1721 from <https://nvd.nist.gov/vuln/detail/CVE-2019-1721> [27]

### 3.2.4 OWASP Top 10

The Open Web Application Security Project, or OWASP, is a non-profit foundation with efforts focused on providing “unbiased, practical, cost-effective information about application security” [28]. Referenced a number of times in this thesis already, OWASP’s primary relationship to this discussion is its release of its OWASP Top Ten Project, a report focusing on the ten most critical web application risks. Considered to be one of the primary categorization systems, the OWASP Top 10 encompasses a large number of the most common and most severe types of attacks against applications, and more specifically, web applications. The OWASP Top 10, most recently published in 2017, is detailed below [29]:

1. *Injection* – A broad categorization, injection covers standard SQL injections, but also attacks such as NoSQL injections and OS injections. These types of attacks are categorized by some untrusted data payload being sent within a command or query maliciously, with the goal of executing unintended commands or revealing sensitive data.
2. *Broken Authentication* – Primarily focusing on session management and user authentication, this set of attacks discusses malicious actors attempting to collect passwords, session tokens or other authentication mechanisms to assume a user or administrator’s identity.

3. *Sensitive Data Exposure* – Web applications and associated APIs offer access to certain pieces of information, but attackers can take advantage of any sensitive information not properly hidden. Valuable and potentially sensitive information can be taken by attackers in transit.
4. *XML External Entities (XXE)* – Discussed earlier in this section, XXE attacks on legacy or weak XML processing entities that can be used to reveal files, access ports on a system or even remote code execution.
5. *Broken Access Control* – Somewhat related to OWASP #2, broken access control focuses on restrictions on user accesses within a system. These types of attacks attempt to exploit flaws in unauthorized access to other accounts, sensitive files or changing rights of other users within a system.
6. *Security Misconfiguration* – The most common security issue found, security misconfiguration describes a number of vulnerabilities. Most commonly, this is found through insecure default configurations, poorly configured HTTP headers, overly verbose error messages and other processes that may contain sensitive information or allow for attacks against poorly configured systems.
7. *Cross-Site Scripting (XSS)* – XSS allows for attackers to execute scripts in a browser window to redirect to malicious sites, imitate real websites or even hijack sessions through poor validation systems. If applications allow untrusted data in a new webpage, XSS attacks can occur.
8. *Insecure Deserialization* – A harbinger of remote code execution, insecure deserialization can be exploited to perform injection and privilege escalation attacks.
9. *Using Components with Known Vulnerabilities* – Fairly self-explanatory, this describes a large set of vulnerabilities created by systems using older components, libraries, software modules and other vulnerable. If an external library has known vulnerabilities and is implemented in the development of an organization's system, it can leave that organization vulnerable to attacks.
10. *Insufficient Logging and Monitoring* – These vulnerabilities often exacerbate other attacks, allowing a malicious actor to take advantage of data or information collected due to poor internal logging and monitoring by an organization. Insufficient logging and monitoring vulnerabilities can be exacerbated by things like plaintext password logging systems and other compounding weaknesses.

The OWASP Top 10 provides an understandable and effective system to classify security risks and the attack mechanisms. As described in the above sections, the CWE can effectively be classified underneath the OWASP Top 10, with a small section for other vulnerability and weakness types. But the OWASP Top 10 is considered by many to be a viable industry standard for efficiently and effectively bucketing vulnerabilities.

### **3.2.5 VulnCat**

Developed by the Fortify Software Security Research Group at HP, VulnCat (or the Fortify Taxonomy) is a categorization system for software security errors. VulnCat is organized

into 7 categories, or phyla, with an homage to the biological kingdoms, indicated in order of importance: (i) Input Validation and Representation, (ii) API Abuse, (iii) Security Features, (iv) Time and State, (v) Errors, (vi) Code Quality and (vii) Encapsulation [30]. The eighth phylum describes security flaws found outside of code itself, or the (viii) Environment that the flaw exists within [30]. This categorization is also briefly discussed in section 4.4.

VulnCat attempts to “organize sets of security rules that can be used to help software developers understand the kinds of errors that have an impact on security” [30]. Primarily developed as a teaching mechanism, VulnCat can be used to educate the software community on the ways that they are unknowingly or unintentionally integrating security concerns into their systems. Each kingdom contains a number of weaknesses. Each weakness is described and provides a detailed explanation, with sample code to describe how it may manifest itself. An example of Access Control Manipulation on AWS S3 buckets is shown in Figure 15. Each weakness is associated with a number of references, all indicating mappings to necessary standards. This weakness in particular is referenced by CWE ID-359 and 287, by the OWASP Top 10, Sensitive Data Exposure, and the PCI DSS (Payment Card Industry Data Security Standards) dating back to PCI v1.1. This type of classification system provides an effective educational tool for software engineers, trying to better understand and solve problems that researchers and practitioners face.

The screenshot shows a page from Fortify Taxonomy titled "Fortify Taxonomy: Software Security Errors". It details a weakness in the "Kingdom: Input Validation and Representation". The specific weakness is "Access Control: ACL Manipulation", which is categorized under "Java/JSP". The abstract states: "The application allows an attacker to manipulate the Access Control Policy (ACP) on an AWS S3 bucket or object." The explanation states: "The application allows an attacker to control the permissions granted to an AWS S3 bucket or object. This allows the attacker to set a weak policy that can enable them to read the content or write arbitrary files." An example code snippet is provided, showing a Java code snippet that creates a new bucket using an Access Control Policy specified by the user. The code is as follows:

```
String canned_acl = request.getParameter("acl");

CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
    .bucket("foo")
    .acl(canned_acl)
    .createBucketConfiguration(CreateBucketConfiguration.builder().locationConstraint(region.id()).build())
    .build();
```

Even if the application expects the `acl` parameter to be selected from a limited set, an attacker can set it to `public-read-write` and grant full anonymous access to the bucket.

Figure 15: Fortify Taxonomy, ACL Manipulation from [https://vulncat.fortify.com/en/detail?id=desc.dataflow.java.access\\_control.acl\\_manipulation#Java%2fJSP](https://vulncat.fortify.com/en/detail?id=desc.dataflow.java.access_control.acl_manipulation#Java%2fJSP) [30]

### 3.3 Benefits of Classification Systems vs. Databases & Enumerations

The CVE, CWE and NVD provide significant value for the security community. The comprehensiveness and reliability of these three industry standards offer a taxonomy that is significant in size and largely encompassing. Organizations like OWASP, HP's Fortify and ENISA all create and maintain frameworks for analyzing and conveying important security information. Both framework and standard types offer varying levels of educational benefits, industry analyses and business translations from the security industry.

In particular, one of the goals of vulnerability and exploit classifications should be to increase security awareness. For most organizations, a CISO will struggle to present the NVD to employees with the hopes of improving security awareness and fostering stronger defenses against malicious actors. Instead, security experts focus on the OWASP Top 10, VulnCat, ENISA, the SANS Top 20 and other systems to consolidate and convey important information. In the next section of the thesis, two security experts share their thoughts on these classifications and other topics including scoring systems and remediation strategies.

## 4. Views from CISOs and Security Experts

During the research phase of this thesis, two security executives were interviewed for their opinions, recommendations and criticisms of scoring systems and particularly vulnerability reporting taxonomies. Mike Shema, the Head of Product Security at Square, and Joe Sechman, the Head of Penetration Test Delivery at Cobalt, both share their thoughts below.

### 4.1 Introduction of Mike Shema

Currently the Head of Security at Square, Mike has been engrossed in the cybersecurity industry for over 13 years, working on security teams at Qualys, Yahoo, Cobalt and now Square [31]. With in-depth knowledge as CISO and significant experience with penetration testing and organizational security, Mike brings a practical perspective to vulnerability classification, remediation and scoring.

### 4.2 Introduction of Joe Sechman

Now the VP of Security Operations and Penetration Test Delivery at Cobalt, Joe Sechman also brings 13+ years of security experience. Previously working at Hewlett Packard for their Fortify Products, Joe led the implementation of the first unified taxonomy for software vulnerabilities with enterprise security products [32]. A speaker at RSA and former Head of Security Engineering at Philips, Joe's background is thoroughly tied to developing and improving taxonomies for software vulnerabilities. Joe also has background as a security consultant, which offered the opportunity for him to work with security flaws in multiple industry verticals.

### 4.3 Vulnerability Scoring

Both interviews opened with discussions of security scoring systems, with Joe focusing on the business considerations. "The key is understanding how to prioritize, not to fix every single vuln" [32]. Joe's background in consultancy offered practical discussions, focusing on prioritization and quantifying the possible money lost in a hack or conversely money saved by patching properly. Both Mike and Joe offered cautious criticism towards CVSS (v2 and v3). Mike admits that the wider security community realizes the full risk of a vulnerability cannot be quantified and that a universal scoring may not be feasible. However, both Mike and Joe qualify this, acknowledging that systems like CVSS offer relative value scoring, which has merit in its own right.

Independently, both praised Bugcrowd's Vulnerability Rating Taxonomy as a valuable frame of reference for both scoring systems and practical bucketing. Both agree it is not comprehensive enough to be considered a taxonomy or even detailed enough for threat modeling, but praise Bugcrowd's frame of reference and practical approach to scoring. Joe

showed particular interest in Kenna's security prioritization system, as it strongly aligned with his initial points against any attempt to fix all vulnerabilities in a system.

Mike offered an interesting perspective on scoring systems, focusing more on the factors necessitating these systems and why they are so complex. One of the most demanding sectors in the security space, with direct influence on remediation requirements, is cyberinsurance. Mike briefly described the complexities of this field, focused on actuarial analysis of protecting against any damages that may result from an attack or breach in an organization's system. These entities attempt to insure companies in the event of a massive breach or attack on their system. To properly price this insurance, a strong understanding of the company's remediation strategies is required. This is where industry standards like CVSS and other well-formed base scoring systems can add significant value, as a cyberinsurance organization can use these relative values to best acquire a risk score for an entire organization. Not the focus of this paper, cyberinsurance is highly complex, but fundamentally tied to things like vulnerability scoring systems and remediation strategies.

#### 4.4 Taxonomies and Classifications

Both interviews also touched upon topics outside of remediation and scoring, most significantly related to vulnerability taxonomies. Joe, with a background in creating and developing VulnCat (<https://vulncat.fortify.com/en>), HP Fortify's taxonomy of software security errors, was keen on this topic. The goal of developing VulnCat was to ensure accurate and consistent representation of risks, which necessitates a comprehensive taxonomy of errors that create these risks. The system focuses on categories or phyla, the first seven of which focus on security flaws within code: (i) Input Validation and Representation, (ii) API Abuse, (iii) Security Features, (iv) Time and State, (v) Errors, (vi) Code Quality and (vii) Encapsulation. The eighth phylum describes security flaws found outside of code itself, or the (viii) Environment that the flaw exists within. Briefly discussed above in section 3.2.5, VulnCat is a valuable approach to security error classification.

Shifting away from academic taxonomies, Mike focused on practical classifications and categorizations, citing the Australian Cyber Security Centre (ACSC) and their Top 4 Strategies to Mitigate Cyber Security Incidents [33]. A practical approach, these 4 strategies offer an incredibly relevant framework for organizations to mitigate possible security incidents and remediate concerns as effectively as possible. The list focuses on Mitigation Strategies to (i) prevent malware delivery and execution, (ii) limit the extent of cyber security incidents, (iii) detect cyber security incidents and respond, and (iv) recover data and system availability. An impressively comprehensive system, Mike cites the ACSC threat model as place that many CISOs refer back to when attempting to mitigate security concerns.

The four strategies in ACSC are coupled with a fifth: mitigation strategies specific to preventing malicious insiders. This concept relates back to section 2.3: employee entity scoring systems. The ACSC focuses on personnel management and the necessity of organizations to



monitor “malicious intent, development of malicious intent, or carrying out malicious intentions undiscovered until after damage has been done” [33]. The concept of an insider threat is not a quantitative one; it is fundamentally difficult to find a scoring system or taxonomy to easily define it, but from the perspective of Mike and Joe, this is the key to managing security inside their organizations. These concerns are often drivers against outsourcing work or ensuring that outsourced workers are managed properly. Mike draws from his experience at Square, where he describes a hypothetical password recovery scenario. If password management were to be organized through an outsourced customer service department, an entity that is not part of Square, Mike would not be able to fully control the actions of that password reset anymore. Instead, he has to rely on the outsourced employee to exercise discretion when handling sensitive information, which adds layers of insecurity to their entire system. From the perspective of a CISO, performing security screenings of new employees and promptly disabling access for those who leave are essential steps in maintaining a secure organization.

## 5. Business Ramifications and Conclusions

### 5.1 Industry Verticals & Remediation in Practice

This section focuses on how businesses and organizations approach cybersecurity and concerns identified in implementation of these concepts. With respect to remediation strategies, organizations consistently accept sizable sets of unfixed vulnerabilities. As discussed in section 2, this is expected and recognized as acceptable risk, but under the assumption that the unfixed vulnerabilities are those with the lowest severity.

The findings from the Cobalt dataset in section 1 showed only 9% of all vulnerabilities labeled as “not fix”, but industry-wide remediation levels are much lower than 91%. If we look at the averages across industry verticals, we can see the probability of vulnerability remediation after identification of the vulnerability is low and incredibly slow. Figure 16, provided by Kenna Security, represents these trends graphically [7].

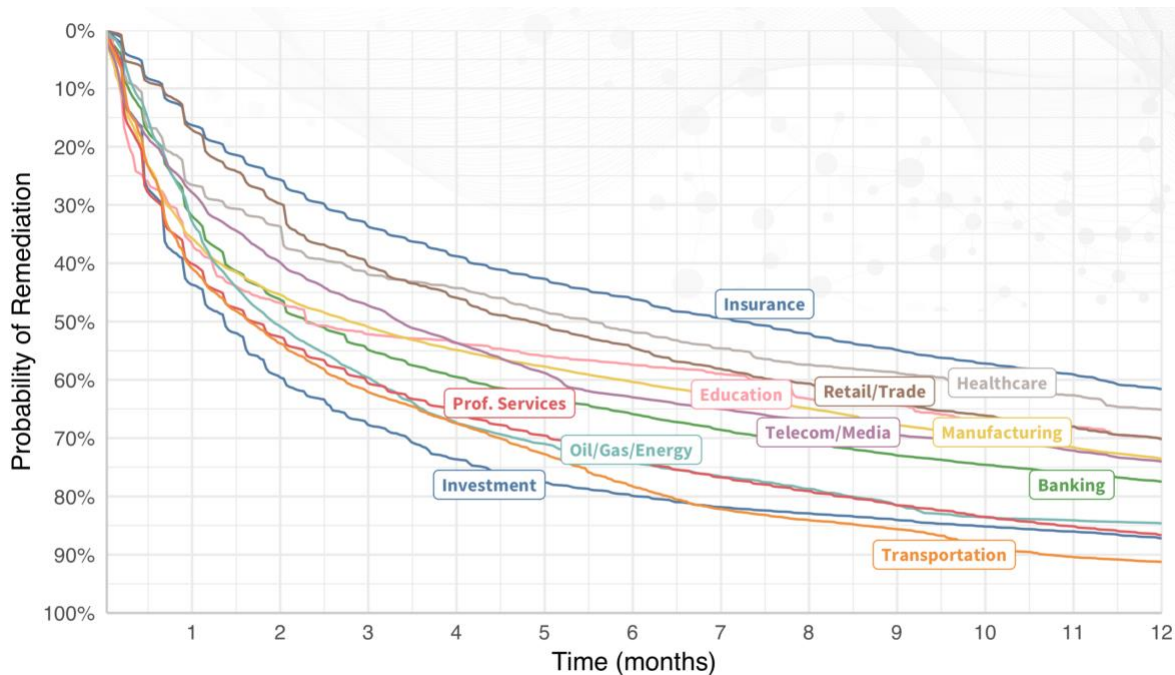


Figure 16: Remediation velocity levels by Industry over time [7]

Focusing on the far-right endpoints of the graph, no industry, save Transportation, reaches 90% remediation likelihood after an entire year. It should be particularly concerning that the probability of a vulnerability fix for the Healthcare sector only occurs with 65% likelihood during the same timeframe. This begs the question: why are so many vulnerabilities not being fixed and why does it take so long to fix them? This thesis has attempted to resolve portions of this question, with efficiency and coverage tradeoffs, but delving a bit deeper into these

statistics indicates that remediation efforts may be worse than initially predicted and the strongest systems we have for vulnerability scoring may not represent the severity of vulnerabilities effectively.

## 5.2 Surprising Findings and Conclusions

### 5.2.1 Do security teams know how to remediate?

Figure 17, also from Kenna Security, focuses on remediation velocity by CVSS value. Of all the conclusions from the Kenna reports, this one sticks out as the most notable. A large majority of vulnerabilities remain unfixed after 12 months, similar to the proportions seen in Figure 16. Acknowledging partial coverage as a reality is appropriate, but only if remediation is effective. Figure 17 indicates that organizations' remediations may not be particularly effective.

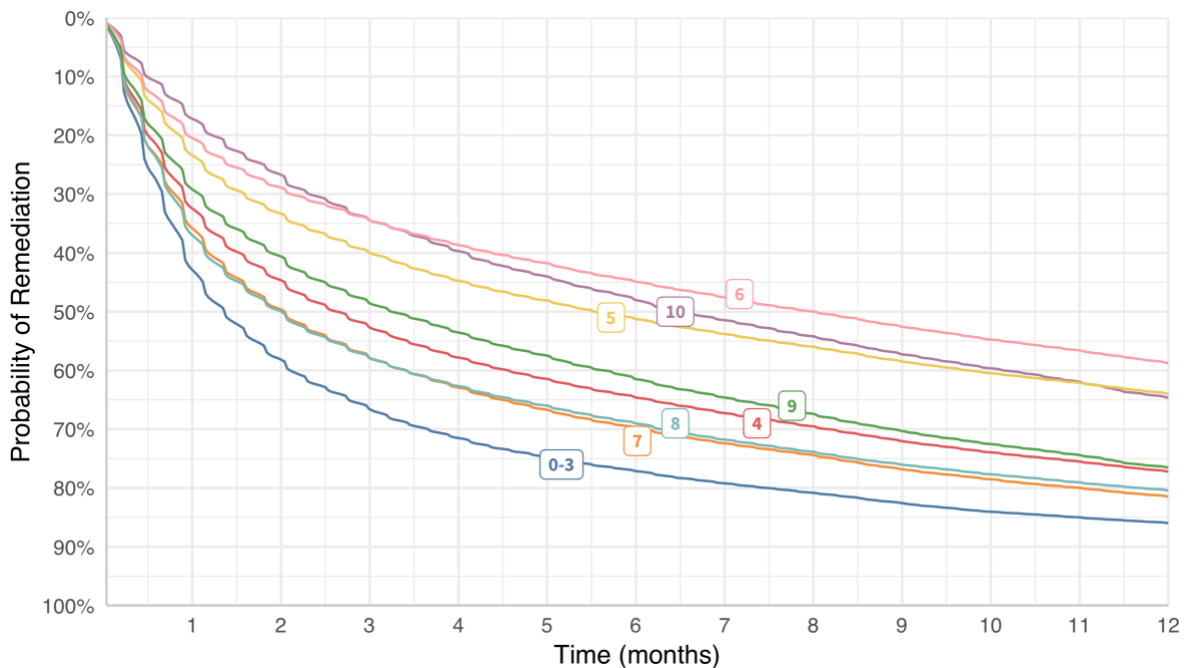


Figure 17: Remediation velocity levels by CVSS over time [7]

What Figure 17 depicts is concerning. Instead of remediating the vulnerabilities identified to be the most severe or critical (CVSS 9.0-10.0), organizations most commonly remediate vulnerabilities that are the *least* severe (CVSS 0.0-3.0). Why would an organization fix the least important vulnerabilities the most often? There are a number of hypotheses to rationalize this trend: (i) CVSS does not reflect severity, (ii) organizations cannot effectively prioritize remediation efforts, (iii) security teams do not know how to remediate more severe vulnerabilities.

- (i) The first conclusion is that CVSS does not properly reflect the severity of vulnerabilities. If this is the case, organizations are superior at identifying the most significant weaknesses in their systems as compared to CVSS. After identifying these weaknesses, they remediate them fairly quickly, with little regard for the CVSS severity value or what other academic descriptions of the vulnerability may be. This would support some concerns previously discussed surrounding CVSS but is a significant deviation from the current posture in the security space.
- (ii) The second conclusion is that organizations do not know how to properly prioritize remediation efforts and are unable to implement the processes described in section 2 of this thesis. The inverse of the first conclusion, this assumes that security teams in organizations struggle to properly identify the most severe vulnerabilities and the CVSS values offer a stronger estimate for the severity of a vulnerability. This would explain why Figure 17 shows so many vulnerabilities with the least severe ratings being fixed at higher and faster rates than those with more severe ratings.
- (iii) The final conclusion assumes that CVSS properly reflects severity of vulnerabilities and that organizations can properly prioritize vulnerabilities. This implies that security teams may identify more severe vulnerabilities, but do not know how to remediate them. This seems to be the most realistic of the three hypotheses, with the assumption that most security teams fix typical, less severe vulnerabilities because they are easily approachable and give teams clear and small victories. This allows more serious weaknesses to fester and increases the likelihood of serious security breaches.

The findings here may indicate that practical remediation and scoring pursuits are less effective than anticipated and difficult to realistically implement. Systems like CVSS, VRT and other remediation bases are undeniably beneficial, but these findings support significant opportunity for improvement in vulnerability scoring and remediation. These hypotheses offer intriguing questions to consider for future research, identified in section 5.3.

## **5.2.2 Conclusions – Vulnerability Scoring and Classification**

While section 5.2.1 may paint the current security climate as bleak, the security systems and taxonomies described in this thesis offer cause for optimism for the future. Unified taxonomies like CVE, CWE and NVD provide comprehensive frameworks for improving universal security, while classification systems like the OWASP Top 10 assist in improving security awareness in organizations. The current state of classification systems in the security space (threat taxonomies, defense-focused systems, vulnerability/weakness enumerations, bucketing systems) is robust and can be effective in assisting the mitigation of hacks and breaches in the future.

Vulnerabilities and weaknesses will never cease to exist in software-based systems. As organizations continue to develop new software, new vulnerabilities and new exploits will emerge. Due to this cycle, refining and reassessing remediation strategies and vulnerability scoring systems will always be a priority for security teams. The systems described in this thesis show promise in identifying vulnerabilities that need remediation, but the value of these processes still lies in the ability of software engineers to make the necessary patches. The focus of the security industry will be faster, more effective patches and increasing global security awareness, with scoring systems and taxonomies as the respective foundations.

### **5.3 Questions for Future Investigation**

Throughout this thesis, there are a number of noted topics not discussed in detail that could be interesting topics for future research. An interesting investigation could focus on the vulnerability scoring systems provided by Bugcrowd and Cobalt, focusing on how hackers tend to report severity scores and optimal strategies to request self-reported weaknesses from them. Elaborating on the importance of security awareness, future research could include a quantitative study on security awareness levels in major companies, focusing on the ability of vulnerability taxonomies to improve awareness. Further topics could include a study into the effectiveness of CVSS and why organizations fail to remediate the seemingly most severe vulnerabilities.

# Appendix A [34]

## 1 CVSS Access Vector Values

Value	Description
<b>Local (L)</b>	The attacker must either have physical access to the vulnerable system (e.g. firewire attacks) or a local account (e.g. a privilege escalation attack).
<b>Adjacent Network (A)</b>	The attacker must have access to the broadcast or collision domain of the vulnerable system (e.g. ARP spoofing, Bluetooth attacks).
<b>Network (N)</b>	The vulnerable interface is working at layer 3 or above of the OSI Network stack. These types of vulnerabilities are often described as remotely exploitable (e.g. a remote buffer overflow in a network service)

## 2 CVSS Access Complexity Values

Value	Description
<b>High (H)</b>	Specialized conditions exist, such as a race condition with a narrow window, or a requirement for social engineering methods that would be readily noticed by knowledgeable people.
<b>Medium (M)</b>	There are some additional requirements for access, such as a limit on the origin of the attacks, or a requirement for the vulnerable system to be running with an uncommon, non-default configuration.
<b>Low (L)</b>	There are no special conditions for access to the vulnerability, such as when the system is available to large numbers of users, or the vulnerable configuration is ubiquitous.

## 3 CVSS Authentication Values

Value	Description
<b>Multiple (M)</b>	Exploitation of the vulnerability requires that the attacker authenticate two or more times, even if the same credentials are used each time.
<b>Single (S)</b>	The attacker must authenticate once in order to exploit the vulnerability.
<b>None (N)</b>	There is no requirement for the attacker to authenticate.

## 4 CVSS Confidentiality Values

Value	Description
<b>None (N)</b>	There is no impact on the confidentiality of the system.
<b>Partial (P)</b>	There is considerable disclosure of information, but the scope of the loss is constrained such that not all of the data is available.
<b>Complete (C)</b>	There is total information disclosure, providing access to any / all data on the system.

## 5 CVSS Integrity Values

<b>Value</b>	<b>Description</b>
<b>None (N)</b>	There is no impact on the integrity of the system.
<b>Partial (P)</b>	Modification of some data or system files is possible, but the scope of the modification is limited.
<b>Complete (C)</b>	There is total loss of integrity; the attacker can modify any files or information on the target system.

## 6 CVSS Availability Values

<b>Value</b>	<b>Description</b>
<b>None (N)</b>	There is no impact on the availability of the system.
<b>Partial (P)</b>	There is reduced performance or loss of some functionality.
<b>Complete (C)</b>	There is total loss of availability of the attacked resource.

---

<sup>1</sup> “Cybersecurity Industry Overview.” *Prime Indexes*, Prime Indexes, [www.primeindexes.com/indexes/prime-cyberindex.html](http://www.primeindexes.com/indexes/prime-cyberindex.html).

<sup>2</sup> Hannula, Tero. “A Framework for Securing Internal Business-Critical Infrastructure Services.” *Institute of Information Technology*, Degree Programme in Cyber Security, 2018.

<sup>3</sup> Houser, Adam Michael. “Mental Models for Cybersecurity: A Formal Methods Approach.” *University at Buffalo, State University of New York*, University at Buffalo, State University of New York, 2018.

<sup>4</sup> “Addressing the SANS Top 20 Critical Security Controls for Effective Cyber Defense.” *Trend Micro*, Feb. 2016, [resources.trendmicro.com/rs/945-CXD-062/images/sans\\_top20\\_csc\\_trendmicro2016.pdf](https://resources.trendmicro.com/rs/945-CXD-062/images/sans_top20_csc_trendmicro2016.pdf).

<sup>5</sup> Srivastava, Amit Kumar, and Shishir Kumar. “An Effective Computational Technique for Taxonomic Position of Security Vulnerability in Software Development.” *Journal of Computational Science*, vol. 25, Mar. 2018, pp. 388–396., doi:10.1016/j.jocs.2017.08.003.

<sup>6</sup> “Prioritization to Prediction, Volume 2 - Getting Real About Remediation.” *Prioritization to Prediction, Volume 2 / Kenna Security*, Kenna Security, 22 Jan. 2019, [resources.kennasecurity.com/research-reports/prioritization-to-prediction-getting-real-about-remediation](https://resources.kennasecurity.com/research-reports/prioritization-to-prediction-getting-real-about-remediation).

<sup>7</sup> “Prioritization to Prediction, Volume 3 - Winning the Remediation Race.” *Prioritization to Prediction, Volume 3 / Kenna Security*, Kenna Security, 12 Mar. 2019, [resources.kennasecurity.com/research-reports-3/prioritization-to-prediction-winning-the-remediation-race-2](https://resources.kennasecurity.com/research-reports-3/prioritization-to-prediction-winning-the-remediation-race-2).

<sup>8</sup> *SecOps Vulnerability Memorandum: Vulnerability Taxonomy, Reporting and Analysis*.

<sup>9</sup> *SecOps Vulnerability Memorandum: Vulnerability Reporting and Analysis*.

<sup>10</sup> “OWASP Top 10 - 2017.” *OWASP.org*, OWASP, 2017, [www.owasp.org/images/7/72/OWASP\\_Top\\_10-2017\\_%28en%29.pdf.pdf](http://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf).

<sup>11</sup> Srivastava, Amit Kumar, and Shishir Kumar. “An Effective Computational Technique for Taxonomic Position of Security Vulnerability in Software Development.” *Journal of Computational Science*, vol. 25, Mar. 2018, pp. 388–396., doi:10.1016/j.jocs.2017.08.003.

<sup>12</sup> Kelleher, Kevin. “Facebook Loses Around \$13 Billion in Value After Data Breach Affects 50 Million of Its Users.” *Fortune*, 28 Sept. 2018, [fortune.com/2018/09/28/facebook-stock-falls-after-security-breach/](https://fortune.com/2018/09/28/facebook-stock-falls-after-security-breach/).

<sup>13</sup> “Prioritization to Prediction Report, Volume 1.” *Prioritization to Prediction Report | Kenna Security*, Kenna Security, 24 Aug. 2018, [resources.kennasecurity.com/research-reports/prioritization-to-prediction](https://resources.kennasecurity.com/research-reports/prioritization-to-prediction).

<sup>14</sup> Mell, Peter, et al. “CVSS v2 Complete Documentation.” *FIRST*, [www.first.org/cvss/v2/guide](http://www.first.org/cvss/v2/guide).

<sup>15</sup> Poonia, Ajeet Singh, et al. “Vulnerability Identification and Misuse Case Classification Framework.” *Advances in Intelligent Systems and Computing Soft Computing: Theories and Applications*, 25 Nov. 2017, pp. 659–666., doi:10.1007/978-981-10-5699-4\_62.

<sup>16</sup> “Bugcrowd's Vulnerability Rating Taxonomy.” *Bugcrowd*, 13 Mar. 2019, [bugcrowd.com/vulnerability-rating-taxonomy](https://bugcrowd.com/vulnerability-rating-taxonomy).

<sup>17</sup> “Cobalt Application Security Platform.” *Cobalt*, 2019, [cobalt.io/how](https://cobalt.io/how).



- 
- <sup>18</sup> “Prioritization to Prediction Report, Volume 1.” *Prioritization to Prediction Report | Kenna Security*, Kenna Security, 24 Aug. 2018, [resources.kennasecurity.com/research-reports/prioritization-to-prediction](https://resources.kennasecurity.com/research-reports/prioritization-to-prediction).
- <sup>19</sup> Ani, Uchenna P. Daniel, et al. “Human Capability Evaluation Approach for Cyber Security in Critical Industrial Infrastructure.” *Advances in Intelligent Systems and Computing Advances in Human Factors in Cybersecurity*, vol. 501, 10 July 2016, pp. 169–182., doi:10.1007/978-3-319-41932-9\_14.
- <sup>20</sup> “2018 Data Breach Investigations Report 11th Edition.” *Verizon.com*, Verizon, 2018, [enterprise.verizon.com/resources/reports/2018/DBIR\\_2018\\_Report.pdf](https://enterprise.verizon.com/resources/reports/2018/DBIR_2018_Report.pdf).
- <sup>21</sup> “Common Weakness Enumeration - CWE-611.” *CWE*, 3 Jan. 2019, [cwe.mitre.org/data/definitions/611.html](https://cwe.mitre.org/data/definitions/611.html).
- <sup>22</sup> “ENISA Threat Landscape Report 2018.” *ENISA*, 15 Feb. 2019, [www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018](http://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018).
- <sup>23</sup> Hannula, Tero. “A Framework for Securing Internal Business-Critical Infrastructure Services.” *Institute of Information Technology*, Degree Programme in Cyber Security, 2018.
- <sup>24</sup> “Common Weakness Enumeration.” *CWE*, [cwe.mitre.org/about/index.html](https://cwe.mitre.org/about/index.html).
- <sup>25</sup> “Common Vulnerabilities and Exposures (CVE).” *CVE*, [cve.mitre.org/](https://cve.mitre.org/).
- <sup>26</sup> Li, Frank, and Vern Paxson. “A Large-Scale Empirical Study of Security Patches.” [Http://People.eecs.berkeley.edu](http://People.eecs.berkeley.edu), University of California, Berkeley and International Computer Science Institute, [people.eecs.berkeley.edu/~frankli/papers/li-ccs2017.pdf](http://people.eecs.berkeley.edu/~frankli/papers/li-ccs2017.pdf).
- <sup>27</sup> “National Vulnerability Database.” *NVD*, NIST, [nvd.nist.gov/](https://nvd.nist.gov/).
- <sup>28</sup> “Category:OWASP Application Security Verification Standard Project.” *OWASP*, [www.owasp.org/index.php/Category:OWASP\\_Application\\_Security\\_Verification\\_Standard\\_Project](http://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project).
- <sup>29</sup> “OWASP Top 10 - 2017.” *OWASP.org*, OWASP, 2017, [www.owasp.org/images/7/72/OWASP\\_Top\\_10-2017\\_%28en%29.pdf.pdf](http://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf).
- <sup>30</sup> “Fortify Taxonomy: Software Security Errors.” *A Taxonomy of Coding Errors That Affect Security*, MicroFocus, 2019, [vulncat.fortify.com/en](https://vulncat.fortify.com/en).
- <sup>31</sup> Friedman, Jacob. “Interview with Mike Shema (Head of Product Security at Square).” 5 Apr. 2019.
- <sup>32</sup> Friedman, Jacob. “Interview with Joe Sechman (VP of Pen Test Delivery at Cobalt.io).” 11 Feb. 2019.
- <sup>33</sup> “Strategies to Mitigate Cyber Security Incidents – Mitigation Details.” *Australian Signals Directorate*, Australian Cyber Security Centre, Feb. 2017, [www.cyber.gov.au/sites/default/files/2019-03/Mitigation\\_Strategies\\_2017\\_Details\\_0.pdf](http://www.cyber.gov.au/sites/default/files/2019-03/Mitigation_Strategies_2017_Details_0.pdf).
- <sup>34</sup> Khan, Shakila A. “Analysis of Cybersecurity Vulnerability Trends and Forecast Modeling.” *George Washington University*, ProQuest, 2018.