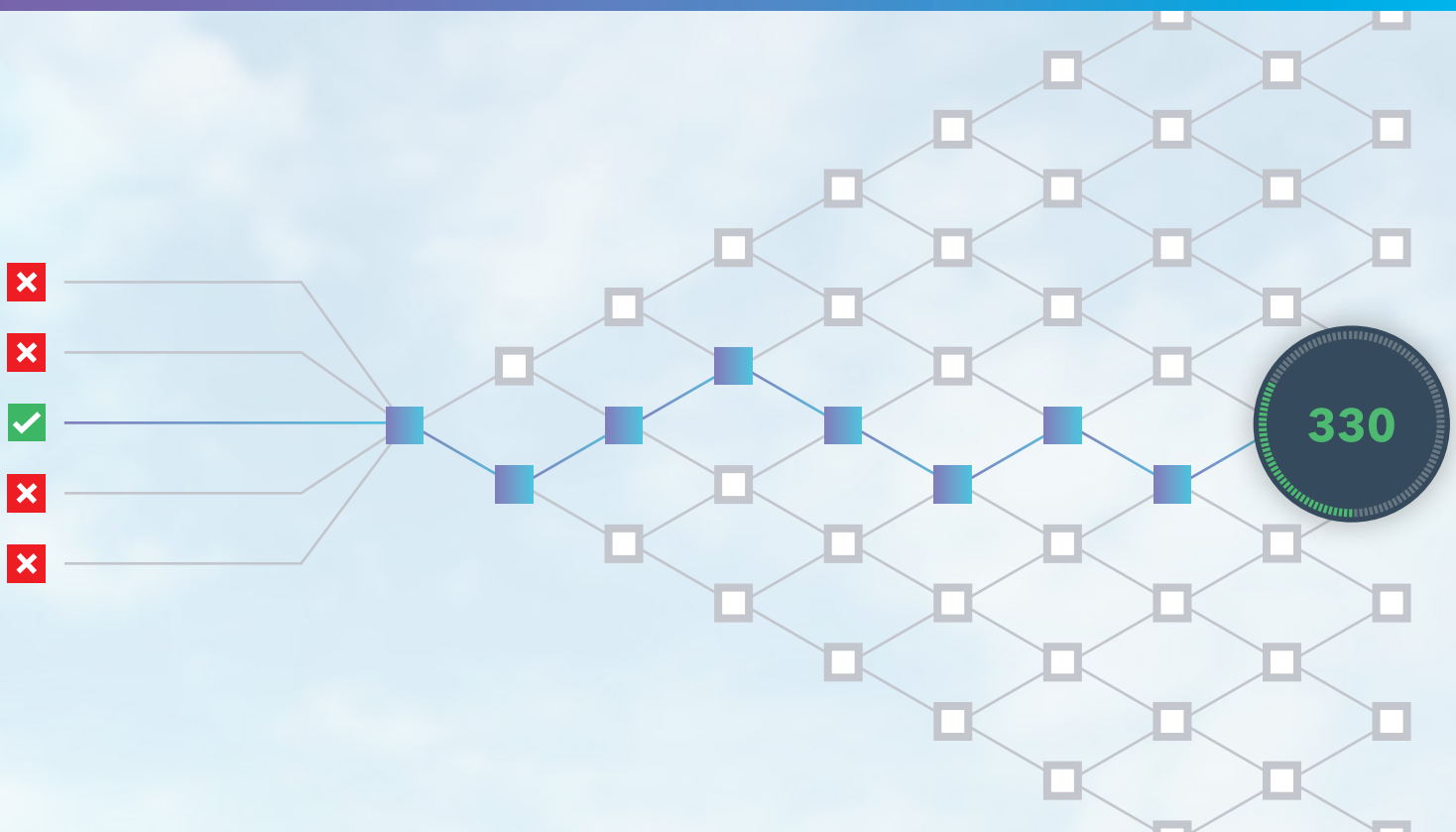
KENNA Security

CYENTIA INSTITUTE

PRIORITIZATION TO PREDICTION

Analyzing Vulnerability Remediation Strategies



PRIORITIZATION TO PREDICTION

KENNA Security

This research was commissioned by Kenna Security. Kenna collected and provided the datasets described herein to the Cyentia Institute for independent analysis and drafting of this report.

Kenna Security is a leader in predictive cyber risk. The Kenna Security Platform enables organizations to work cross-functionally to determine and remediate cyber risks. Kenna leverages Cyber Risk Context Technology™ to track and predict real-world exploitations, focusing security teams on what matters most. Headquartered in San Francisco, Kenna counts among its customers many Fortune 100 companies, and serves nearly every major vertical. For more information, visit www.kennasecurity.com

CYENTIA® INSTITUTE

Analysis for this report was provided by the Cyentia Institute. Cyentia seeks to advance cybersecurity knowledge and practice through data-driven research. We curate knowledge for the community, partner with vendors to create analytical reports like this one, and help enterprises gain insight from their data.

Find out more: www.cyentia.com.

Introduction

To remediate, or not to remediate, that is the question: Whether 'tis nobler in the mind to suffer the slings and arrows of malicious fortune, or to take arms against a sea of vulnerabilities, and by opposing end them.

Who knew that by changing just three words in a famous Shakespeare soliloquy, it would all of a sudden become eerily relevant to cybersecurity? Then again, those who have suffered the Internet's continual volley of 'slings and arrows' or barely kept from drowning in an endless 'sea of vulnerabilities' may not struggle with the exact same questions as Prince Hamlet—but they can feel his pain.

Indeed, few would argue that the relentless—and often haphazard—process of identifying, tracking, prioritizing, and remediating security vulnerabilities ranks among the top 'malicious fortunes' of our profession. It's something every security program must do, but none of them enjoy it, and few do it well. The result is a patchwork of fixes applied to a tide of vulnerabilities, any one of which could be the single point of failure in an otherwise formidable defense.

This means one of the biggest challenges in vulnerability management relates to prioritization, and ideally, prediction. Given the long and growing list of open vulnerabilities that must either be dealt with or delayed, which ones are most likely to be exploited, and thus deserving of priority remediation? Answering this question in the form of a predictive model based on numerous sources of historical data is the goal of this study.

We begin with a review of data sources available for building or improving decision models for vulnerability remediation. We then discuss the vulnerability lifecycle and examine timelines and triggers surrounding key milestones. Identifying attributes of vulnerabilities that correlate with exploitation comes next on the docket. The last section measures the outcomes of several remediation strategies and develops a model that optimizes overall effectiveness.

Thank you for your interest in this research. We sincerely hope our findings will help you take up arms against that sea of vulnerabilities in your organization, and by opposing, end them.

Key Findings

23% of published vulnerabilities have associated exploit code.

2% of published vulnerabilities have observed exploits in the wild.

The chance of a vulnerability being exploited in the wild is 7X higher when exploit code exists.

50% of exploits publish within two weeks surrounding new vulnerabilities.

13% of exploits emerge a month or more after vulnerabilities publish. Only 1% emerge beyond 1 year.

The volume of exploitation detections jumps five-fold upon release of exploit code.

Remediation strategies can be measured in terms of coverage (recall) and efficiency (precision).

Remediating vulnerabilities with CVSS 7+ achieves efficiency of 32% and coverage of 53%.

Remediating vulnerabilities for the top 20 vendors achieves efficiency of 12% and coverage of 22%.

The proposed prediction model performs 2x and 8x more efficiently (respectively) than the approaches above with equivalent coverage.

Data Sources

This study focuses on the vulnerabilities described in MITRE's Common Vulnerabilities and Exposures (CVE) List. But in order to provide more context to the study and help measure the importance of remediating any specific CVE, we also leverage several other sources . We describe these sources and attributes in this section.

MITRE's Common Vulnerabilities and Exposures (CVE)

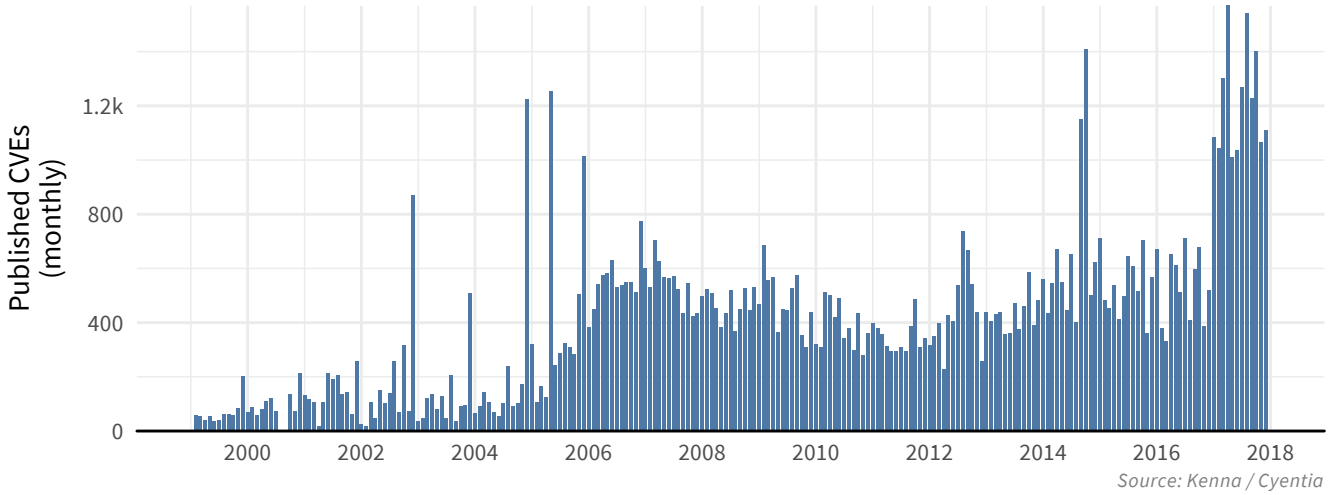
We focus our research on discovered and disclosed vulnerabilities contained in the CVE List from MITRE. We do this primarily because CVEs are publicly tracked, readily available, extensive (although not exhaustive), and have become the de facto standard adopted by many other projects and products. It should be noted, however, that CVEs are neither comprehensive or perfect. Many vulnerabilities are unknown, undisclosed, or otherwise have not been assigned a CVE ID. Furthermore, CVE listings are curated by humans, which makes them vulnerable to biases, errors, and omissions.[1] Despite these challenges, the CVE List is a valuable community resource that greatly assists the otherwise untenable task of vulnerability management.

Between its inception and up to January 1st, 2018, over 120,000 CVE entries have been created. 21,136 of those are still in "reserved" status, meaning they have been allocated or reserved for use by a CNA or researcher, but the details have not yet been populated. 4,352 have been rejected for various reasons and another 8 are split out or labeled as unverifiable. We chose to include the 491 published CVE's currently in the "disputed" state since most describe weaknesses that would be useful to an attacker. A total of 94,597 CVEs from the CVE List were utilized in this research.

> The CVE List is neither comprehensive or perfect. Many vulnerabilities are unknown, undisclosed, or otherwise never assigned a CVE ID. Furthermore, CVE details are subject to biases, errors, and omissions. Even so, it is a valuable project.

For all intents and purposes, each of these published CVEs represents a decision and potential action for vulnerability management programs. The criteria for those decisions may be simple in the singular case (e.g., "Does that exist in our environment?"), but prove to be quite difficult in the aggregate (e.g. "Where do we start?"). Figure 1 reinforces this challenge by demonstrating the increasing volume of reserved and published CVEs over time.

FIGURE 1
Volume of published CVE's from 1999 through 2017



Source: Kenna / Cyentia

[1] *For discussion of these biases and other CVE-related issues, see 2013 BlackHat presentation titled "Buying into the Bias: Why Vulnerability Statistics Suck" from Brian Martin and Steve Christy.*

CVE Enrichment Projects

In addition to the basic CVE information produced by MITRE, this research also leverages the details added to each CVE by the National Vulnerability Database (NVD). NVD enriches the base CVE information with details leveraging other community projects, which include the following:

Common Vulnerability Scoring System (CVSS). This provides a process to capture the principal characteristics of a vulnerability and produce a numerical score that reflects its severity. The CVSS standard has very recently moved to version 3, but the majority of published CVE's were recorded using version 2, so we use version 2 in this report. CVSS was developed and is maintained by the Forum of Incident Response and Security Teams (FIRST).

Common Platform Enumeration (CPE). This provides a standard machine-readable format for encoding names of IT products, platforms and vendors. It was developed at MITRE, but ongoing development and maintenance is now handled by NIST.

Common Weakness Enumeration (CWE) - This provides a common language for describing software security weaknesses in architecture, design, or code. It was developed and is maintained by MITRE. We won't be discussing CWE's in this study.

Each piece of enrichment data offers potentially useful context for decisions. Basic remediation strategies may rely on CVSS alone, while others will factor in the type of vulnerability (CWE) along with the vendor and product and the exposure of the vulnerabilities across environments.

Exploit Code and Observations

Basic analysis of CVE's may stop at data from MITRE and NVD, but those miss an important part of the equation: what CVE's are attackers actually exploiting in the wild? Unfortunately, no universal source of exploit activity exists, so we have to collect this information through multiple direct and indirect ways. These include host and network-based detection systems as well as by reverse engineering the malware and tools used by attackers.

Sources used in this study to track which CVE's have been actively exploited include the SANS Internet Storm Center (monthly statistics from the ISC signature collection honeynetproject), Secureworks CTU (active campaigns associated with CVEs), Alienvaults OSSIM metadata (Reputation feed, collecting IDS signature hits across 100,000+ devices in 150+ countries) and Reversing Labs metadata.

But sometimes observing exploitation in the wild comes too late for risk-averse vulnerability remediation strategies. In such cases, published exploit code serves as a good indicator of exploitability because it enables attackers to easily weaponize a vulnerability. Roughly two out of every three CVEs with active exploit detections also have published exploit code. Tracking the publication of exploit code, therefore, is important to remediation prioritization.

Sources used in this study to track which CVE's have public exploit code include Exploit DB, several exploitation frameworks (Metasploit, D2 Security's Elliot Kit and Canvas Exploitation Framework), the Contagio dump and data from Reversing Labs and Secureworks CTU.

Not only are exploit code releases strongly correlated with active exploitations, but they also indicate something more: the characteristics of a vulnerability that exploit writers target. Even if we haven't seen a specific CVE with published exploit code, the exploited vulnerabilities tend to share similar characteristics and traits with written exploits.

> Roughly two out of every three exploited CVEs have associated published code. When exploit code is published, the chance we observe exploitation in the wild is seven times higher than without published exploit code.

The Vulnerability Lifecycle

Our goal with all of this is to remediate vulnerabilities in the most efficient way. But before we discuss prioritization strategies and models, we first need to establish some basics. Let's start with the meaning of "vulnerability". The term has wide-ranging usage, including general weaknesses in security posture, various types of code-level flaws, and specific entries in the CVE List and the NVD. In general, a vulnerability will hit some combination of the following milestones during its lifecycle:

CREATED - This appears obvious, but is worth stating nonetheless. A vulnerability is created when flawed code is written and released in a vulnerable state. It has the potential for exploitation regardless of whether or not it has been discovered, disclosed, or developed into exploit code.

DISCOVERY - The means and statistics surrounding vulnerability discovery are not the focus of this research. Suffice it to say that when someone learns that a vulnerability exists, it has been discovered.

DISCLOSURE - When a vulnerability is discovered, it may or may not be publicly reported. Vulnerability disclosure is a hot topic in our industry, but again, it is not the focus of our current research. We can only study vulnerabilities that are both discovered and disclosed. For our purposes, "disclosure" implies that a vulnerability has been reported to CVE Numbering Authorities (CNA) and assigned a CVE ID. This puts it on our (and many others') radar for possible remediation. But it should be acknowledged that other sources of vulnerability disclosures exist.

EXPLOIT CODE - When a vulnerability becomes public, proof-of-concept or working code for exploiting it may be published as well. Exploits can be traded in the digital underground, shared on above-ground mailing lists, published in blogs, or included in exploitation frameworks. Vulnerabilities reaching this stage are more exploitable, but still may not be used for actual exploitation.

EXPLOITATION - Attacks targeting a vulnerability in the wild constitute exploitation. The success of those attacks is ultimately what vulnerability management programs work to prevent. The good news (if we can call it that) is that not all organizations are targeted at once. So observing exploits may (and should) escalate remediation priority.

DETECTION SIGNATURE - In order to detect the exploitation of a vulnerability, sensors need to know what to look for. Most tools maintain some form of detection signatures to accomplish this. Unfortunately, these take time to create and distribute, meaning most sensors aren't capable of detecting exploitation until a) a working exploit is published and b) signatures have been distributed. Of course, some exploits will trigger existing or generic signatures (SQL injection is rather universal).

Not all vulnerabilities follow all the milestones in the order presented above. While the majority of CVEs never stray beyond disclosure, some jump straight to active exploitation in the wild. But let's not get ahead of ourselves—a true appreciation of the prioritization challenge in vulnerability management begins with the volume of CVEs published.

CHALLENGES TO VULNERABILITY DATA COLLECTION

We'd be remiss if we didn't acknowledge and discuss some of the issues involved in studying vulnerabilities and exploits. One of the challenges with observing exploitation "in the wild" is that a detection signature has to written and implemented across sensors. In some cases, such as SQL injection or XSS, the signatures and detection should work independent of the specific underlying vulnerability. But some CVE's are unique enough to require their own signature. Generally speaking, it's easier to create a signature using exploit code as a template, so detection of exploitation is often dependent on published exploits. Also, how the detection is done matters a lot. Using network-based detection mechanisms, we may never see host-specific exploits. Additionally, detecting the exploitation of something like CVE-2017-14937 (the detonation of passenger airbags in some cars), opens up a whole new realm of challenges. But think of these numbers as a snapshot in time pieced together from multiple perspectives. We may not have a full panoramic vista in fine detail, but we have enough to examine and improve upon the status quo as the practice of vulnerability management improves.

The Case for Prioritization

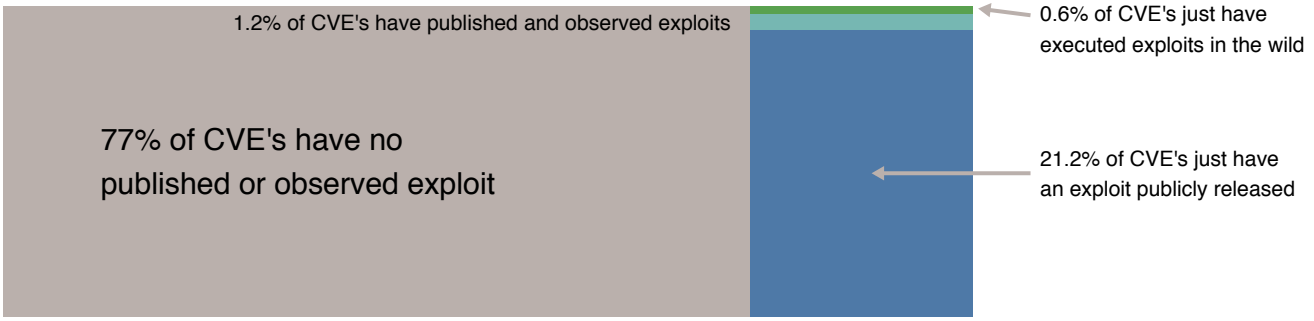
The goal of a vulnerability management program is to remediate vulnerabilities in a cost-effective manner before they lead to security incidents. Remediation may involve deploying updates and patches, modifying system configuration, implementing compensating controls, and a range of other options. But which vulnerabilities warrant this kind of treatment?

Circling back to our earlier statement that the majority of CVEs never proceed beyond disclosure, Figure 2 puts some numbers around that assertion. 77% of CVEs, to be precise, have no exploit code or observed exploitations associated with them. Exploit code has been published for 22% of CVEs at the time of this writing. The fact that an exploit exists does nothing to harm vulnerable assets directly. It does, however, represent a tool that can be used by those who intend to cause harm,[2] which is why exploits are an important factor in remediation decisions. Active exploitation is another risk-altering event. According to our data sources, less than 2% of CVEs have been actively exploited in the wild.

What is not be apparent directly from Figure 2 is the correlation among these milestones. Roughly two out of every three exploited CVEs have associated published code. When exploit code is published, the chance we observe exploitation in the wild is seven times higher than without published exploit code. We refer to vulnerabilities for which either exploit code or active exploitations exist as "exploit exists" for many figures in this report.

FIGURE 2

Comparison of CVEs with exploit code and/or observed exploits in the wild relative to all published CVEs



Source: Kenna / Cyentia

Figure 2 has two important implications, which hold even if we allow a substantial margin of error around these statistics owing to the challenges and biases highlighted previously. First, we have an efficiency problem. Remediation must be cost-effective, and Figure 2 clearly illustrates that remediating either comprehensively or randomly would waste a great deal of resources. Second, we have a coverage problem. If the focus becomes too narrow or is misplaced, the security program will fail in its mission to remediate the CVEs that are most likely to result in costly security incidents.

Successful vulnerability management, then, balances the two opposing goals of coverage (fix everything that matters) and efficiency (delay/deprioritize what doesn't matter). This is the crux of the remediation prioritization challenge and the goal of the prediction model we present later in this report.

In driving toward that goal, we examine an array of attributes, assessments, and prioritization strategies. How quickly are things being exploited? How do decisions based on CVSS score thresholds perform? Should vendors be factored into the model? Does attention within a vulnerability discussion forum portend exploitation?

One more point bears mention before we begin finding answers to those questions. Figure 2 is backward-looking, but a prediction model is inherently forward-looking. In other words, we must be able to identify CVEs as likely candidates for exploitation even though they have not yet been targeted by attacks or developed into exploit code. Sounds like a fun challenge; let's get started.

[2] *Exploits code can be used for good and helpful purposes too. We are not implying that exploit = evil.*

Finding the Balance: Coverage and Efficiency

Deciding which vulnerabilities to remediate is a daunting task. In a perfect world, all vulnerabilities would be remediated as they were discovered, but unfortunately that doesn't happen in reality. With thousands of new vulnerabilities every year multiplied across disparate assets, reality necessitates prioritization. It comes down to choosing a subset of vulnerabilities to focus on first. But how can we measure the quality of prioritization?

There are many different measurement techniques for a binary decision (to remediate or not to remediate). It's tempting to go for overall accuracy—proportion decided correctly—but this can be a little misleading when so many vulnerabilities are never exploited. For example, if a company chooses to never remediate anything, that decision has an accuracy somewhere around 77%, according to Figure 2. It might look like a good strategy on paper, but not so much in practice. Instead of decision model accuracy, we will focus on the two concepts of **COVERAGE** and **EFFICIENCY**.

COVERAGE MEASURES THE COMPLETENESS OF REMEDIATION. Of all vulnerabilities that should be remediated, what percentage was correctly identified for remediation? For example, if 100 vulnerabilities have existing exploits, and yet only 15 of those are remediated, the coverage of this prioritization strategy is 15%. The other 85% represents unremediated risk. Technically, coverage is the true positives divided by the sum of the true positives and false negatives.

EFFICIENCY MEASURES THE PRECISION OF REMEDIATION. Of all vulnerabilities identified for remediation, what percentage should have been remediated? For example, if we remediate 100 vulnerabilities, yet only 15 of those are ever exploited, the efficiency of this prioritization strategy is 15%. The other 85% represents resources that may have been more productive elsewhere. Technically, efficiency is the true positives divided by the sum of the true positives and false positives.

Ideally, we'd love a remediation strategy that achieves 100% coverage and 100% efficiency. But in reality, a direct trade-off exists between the two. A strategy that prioritizes only the "really bad" CVEs for remediation (i.e., CVSS 10) may have a good efficiency rating, but this comes at the cost of much lower coverage (many exploited vulnerabilities have a CVSS score of less than 10). Conversely, we could improve coverage by remediating CVSS 6 and above, but efficiency would drop due to chasing down CVEs that were never exploited.

Timelines of Exploition

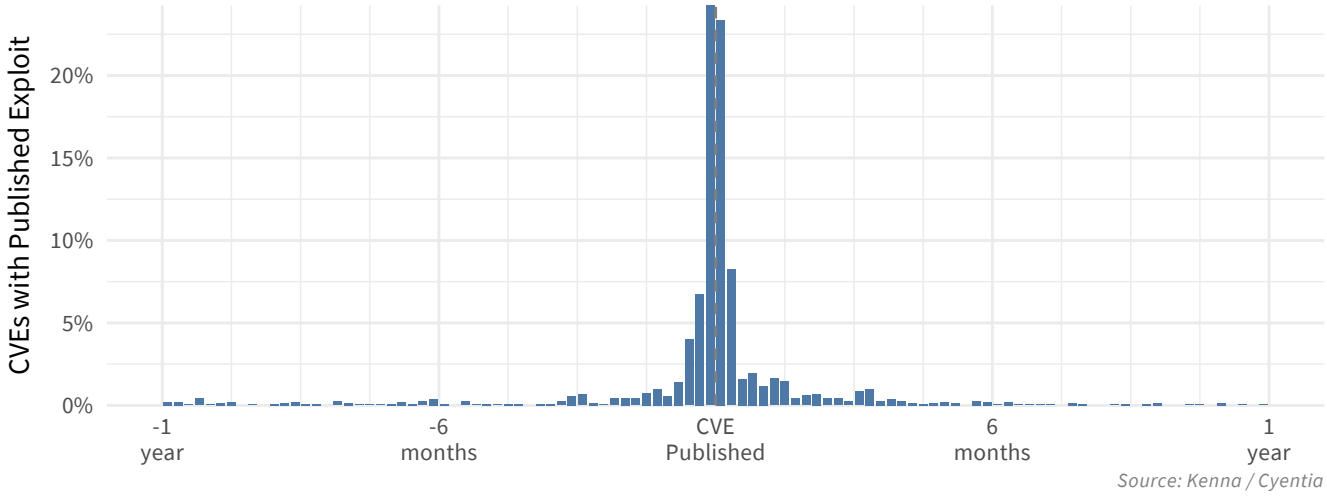
From the previous section, it is evident that vulnerability remediation decisions must balance coverage and efficiency. But it is not yet apparent whether the expediency of these decisions is a critical factor as well. How much time and research can we reasonably devote to deciding whether a vulnerability warrants remediation without risking exposure as exploit code is released and/or malicious exploitation begins? We explore that topic in this section.

Exploit Code Publication

Referring back to Figure 2, it appears that public exploit code exists for just over 22% of all vulnerabilities in the CVE List. But which came first—the CVE or the exploit? And what does the timeline around these milestones look like? Figure 3 charts the relationship between CVE and exploit publication dates.

FIGURE 3

Exploit publication date relative to CVE publication date



Source: Kenna / Cyentia

The major takeaway from Figure 3 is that over half of all exploits are released within two weeks (on either side) of the CVE publication date. At the one month mark, that percentage climbs to 70%. The fairly even split between exploits released right before and soon after CVE publication is apparent, but not too much should be made of this distinction. Dating of CVE reservations and publications is not perfect and is also subject to processing delays. This especially applies to the small number of exploits seen in Figure 3 published far in advance of the CVE. Nearly all of those are either obscure exploits or were developed for CVEs that were reserved (assigned an ID) but not officially published until much later. Again, the story here is that remediation decisions will quickly lose relevance the longer they are delayed.

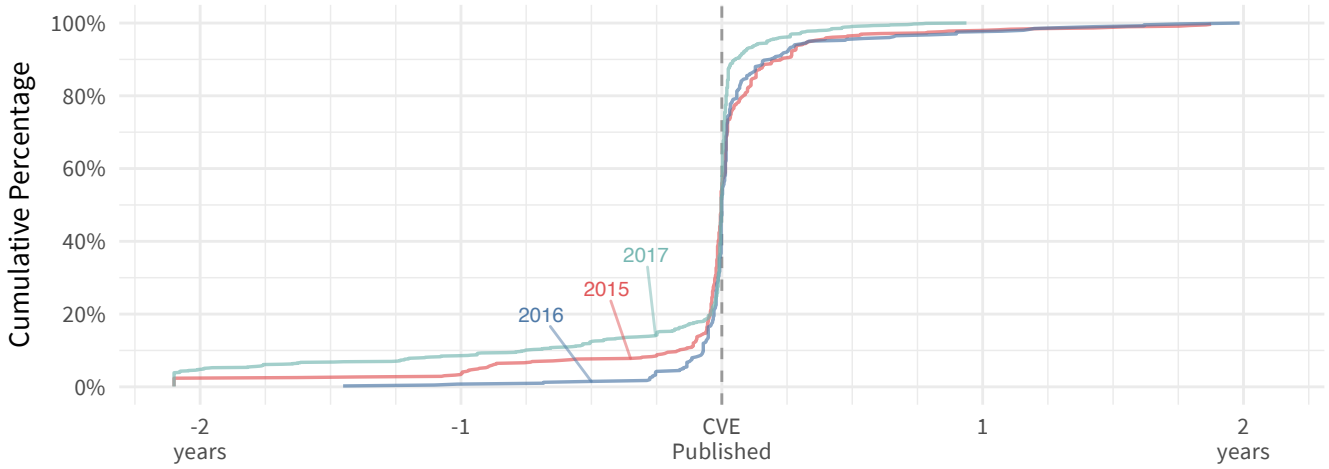
MITRE offers the following disclaimer regarding CVE dating, which places some helpful guardrails around interpretations of statistics in this section:

"The entry creation date may reflect when the CVE ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE."

Figure 4 offers another way to view exploit and CVE publication timelines. It tracks the cumulative percentage of exploit releases leading up to and beyond the publication of associated CVEs in 2015, 2016 and 2017. All three years show a similar trend: a gradual acceleration of exploit development, followed by a major surge immediately surrounding publication, and then a gradual deceleration over time.

FIGURE 4

Exploit publication date relative to CVE *publication* date (Cumulative)



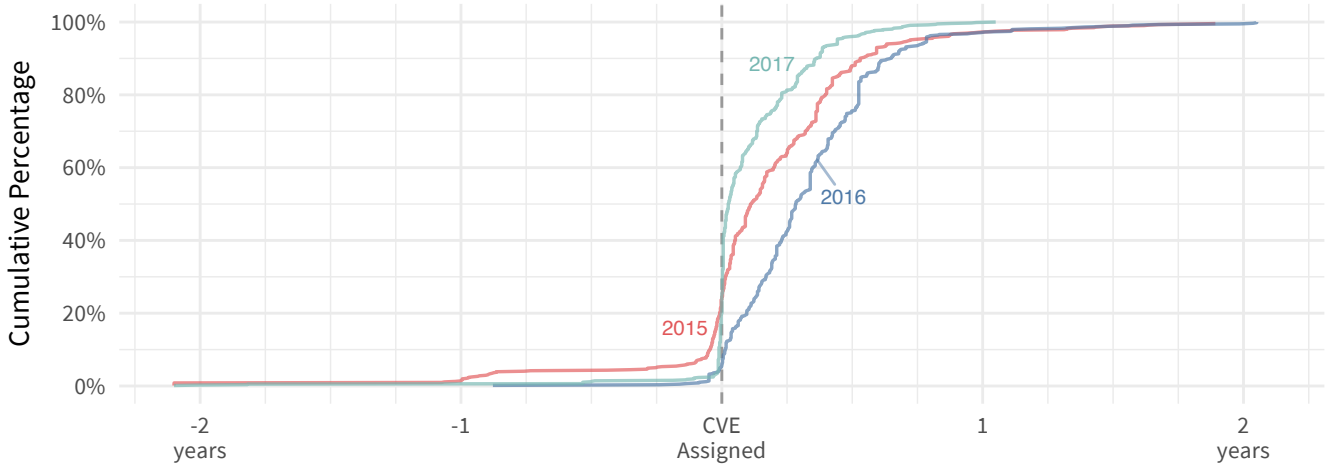
Source: Kenna / Cyentia

In terms of a decision window for remediation decisions, Figures 3 and 4 can be seen as bad news. Both figures above seem to chide "By the time you're aware of a CVE, the tide of battle has already turned and by the time you can make a decision, it'll be lost." But this interpretation is misleading for several reasons.

The first reason has already been mentioned—the CVE publication process suffers from delays and errata. MITRE is working to improve that and there are steps we can adopt to minimize the impact as well. For example, putting CVEs on the remediation radar when they are first reserved or assigned an ID rather than waiting until they are officially published tends to buy some extra time. Figure 5, a redo of Figure 4 using the earliest date on which the CVE ID was assigned rather than the publication date, supports this tactic. Notice how pre-assignment exploit levels are lower and post-assignment acceleration is slower in Figure 5, allowing more time for remediation decisions before exploits become public.

FIGURE 5

Exploit publication date relative to CVE *assigned* date (Cumulative)



Source: Kenna / Cyentia

Another reason we shouldn't give up hope for better decision-making is that not all exploits are created equal. Proof-of-concept code, working or "weaponized" code, and automated exploits incorporated into commodity tools all fall under the common banner of "exploit" in the previous figures. In general, the potential for active exploitation in the wild increases with each of these milestones as exploits become increasingly usable by larger populations of adversaries. This exploit commoditization process sometimes occurs very quickly, and sometimes slowly.

Exploitation in the Wild

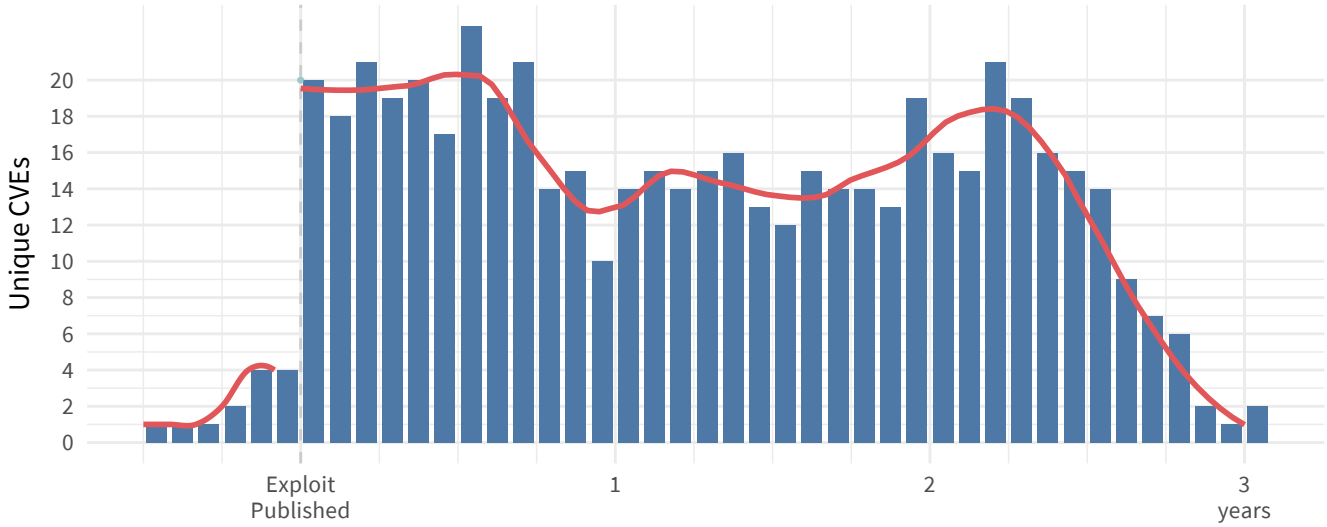
How much time passes between the publication of exploit code and active exploitation? This is the guiding question in this section. As we begin, it is important to touch upon the topic of exploit detection signatures.

In order to detect exploitation, the various tools that perform this function rely on detection signatures (of some form) that essentially specify what to look for at a technical level. These signatures, in turn, rely on information that may not be known until the publication of a working exploit. Once created, signatures must then be distributed to sensors before the associated exploit can be detected. There are more nuances and caveats to this process, but the net result is that the first observation of an exploit in the wild may align more with signature publication than the first instance of exploitation.

This lag may explain the instantaneous jump seen in Figure 6 following the publication of exploit code. Only four unique CVE's show active exploitation prior to that date, but this number catapults five-fold in the month following. The rate of CVEs exploited bounces around over the next two years before slacking off steadily.

FIGURE 6

First detected exploitation relative to exploit publication date, by number of unique CVEs



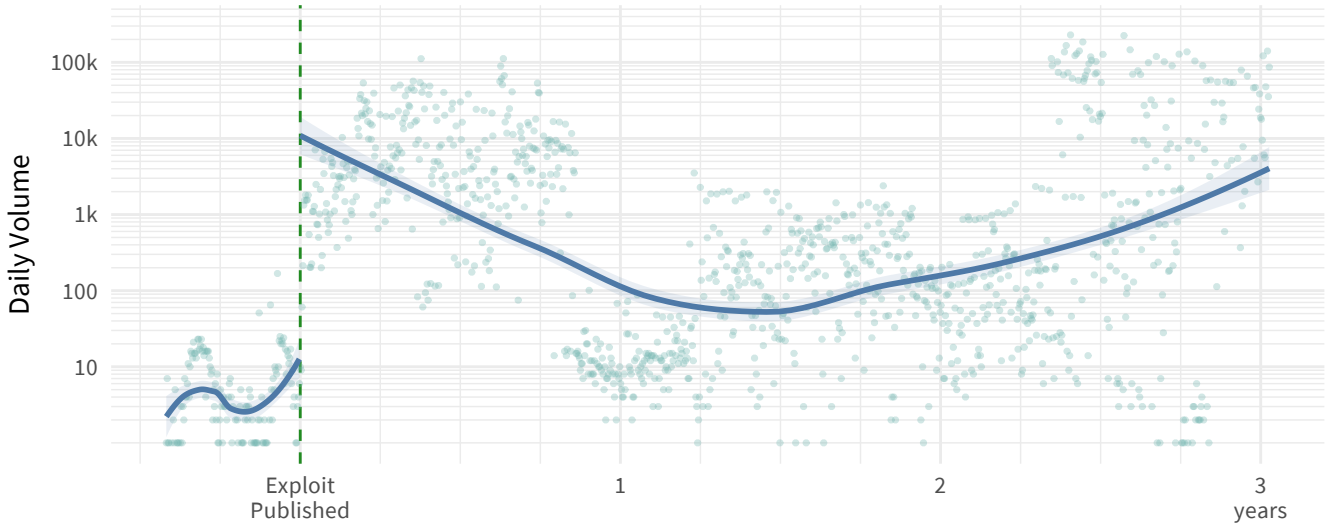
Source: Kenna / Cyentia

Figure 6 is interesting, but isolated observations of exploits targeting CVEs do not correlate strongly with the likelihood that any given organization will be targeted. And that last part is what vulnerability management programs tend to be most concerned with in terms of prioritizing CVEs for remediation. The volume of exploit activity in the wild may be a more suitable gauge of "risk" to organizations represented by CVEs.

Bearing this distinction in mind, we created Figure 7. It plots the daily volume of exploit activity for each CVE along with a trendline running through them (note the log scale on the y-axis). Keep in mind this does not indicate the vulnerability existed in the targeted organization, but rather the triggering of the detection signature(s) associated with a CVE. Similar to Figure 6, we observe limited initial activity, which is followed by a 100 to 1000X burst in volume after publication of exploit code. A clear lull in observations ensues a year after exploit release, and this continues for the next 1.5 years. Curiously, exploit volume trends back up midway through the third year post-publication (for some CVEs, at least).

FIGURE 7

First detected exploitation relative to exploit publication date, by total volume of observations



Source: Kenna / Cyentia

We can make two important inferences from this section regarding the timelines and triggers of vulnerability exploitation. First, exploitation (code and observations) *can* occur simultaneously with (or before) the publication of a CVE, and second, a reasonable chance of exploitation lingers for years. For remediation decisions and models, this further emphasizes the importance of quickly determining vulnerabilities that warrant action and which of those have highest priority. At the same time, vulnerabilities that are not exploited immediately cannot be safely ignored; they must be continually reevaluated along with all the new ones released.

ROLLING THE DICE OF REMEDIATION

While it may seem irrational to prioritize remediation based on random chance, we'll soon discover that some of the simple rule-based strategies do not perform any better than rolling the dice. We will compare various strategies against a "random" approach, which is derived by calculating the expected efficiency and coverage achieved by randomly picking the same number of CVEs to remediate. In this approach, the efficiency would remain constant at about 23%, as that's how many CVEs are labeled as "exploit exists". Coverage, on the other hand, depends on how many CVEs up for remediation, but will grow in proportion to CVE's remediated.

As we evaluate various remediation strategies, we will display this estimated coverage and efficiency from a random model as a point of reference. So, if we test a model that has 23% efficiency and 15% coverage (which is the outcome of remediating all CVEs of CVSS 8 or greater), we can determine whether it beats a dice roll (spoiler alert, it doesn't).

Rules of Remediation

We have established that vulnerability remediation decisions have intense numerical and chronological pressures. But do they have informational challenges as well? Let's see what we can learn about the availability and utility of contextual data about CVEs to support various rule-based remediation strategies.

A description of information available on CVEs ties back to the data sources listed at the beginning of this report. The CVE List itself is rather sparse, with each entry containing an ID, description, and references[3]. The description text fields can be mined to obtain more structured information (and the CVE Team does this), but it is not an ideal source on its own to drive decision models. This is where the NVD and CVE enrichment projects greatly enhance the data we have to work with regarding CVEs. The three main categories we examine in this section include the vendor of the affected product, the CVSS (version 2) score, and other reference lists where researchers share and discuss vulnerabilities.

Vendor Vulnerabilities

Let's get this out of the way before analyzing information related to vendors or products associated with CVEs. *Do not fall into the trap of thinking that more vulnerabilities indicate insecurity.* There are many mundane reasons one vendor may have more CVEs than the next. Perhaps they have high market penetration. Maybe they run bug bounty programs or otherwise encourage the finding and reporting of vulnerabilities. Maybe researchers focus more on certain vendors or products for various reasons. Whatever the rationale, assuming that a comparison of historical CVE counts among vendors is a measurement of (in)security is misleading. Keep that in mind as you study Figure 8.

Figure 8 foregoes the typical "Most Vulnerable Vendors" list in favor of a scatterplot using the count of exploited (x-axis) vs total (y-axis) CVEs. A smoothed regression line adds a visual delineation between vendors above the line that tend to have more CVEs but fewer exploits, and the ones below that have a higher rate of known exploits. While this enables some potentially interesting observations (e.g., Joomla jumps out as unusually exploitable and Google gives the opposite impression), we still should not view this as a proxy for vendor/product security. Figure 8 serves mainly to set up our analysis of a "remediate CVEs from the top X vendors" strategy in Table 1.

One benefit of historical data is that we can use it to test how well different prioritization strategies would have played out. For each CVE, we can apply a strategy, decide if we should remediate it, and then calculate how often that strategy was right and wrong. This allows us to apply and compare the concepts introduced earlier concerning efficiency and coverage.

TABLE 1

Results for prioritization strategies based on vendors with the highest numbers of CVEs

Remediate Vendors	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
Top5	2,598	19,119	18,500	54,380	**12.3%**	**12%**	23%	22.9%
Top10	3,588	18,129	27,705	45,175	**11.5%**	**16.5%**	23%	33.9%
Top20	4,726	16,991	34,471	38,409	**12.1%**	**21.8%**	23%	42.5%

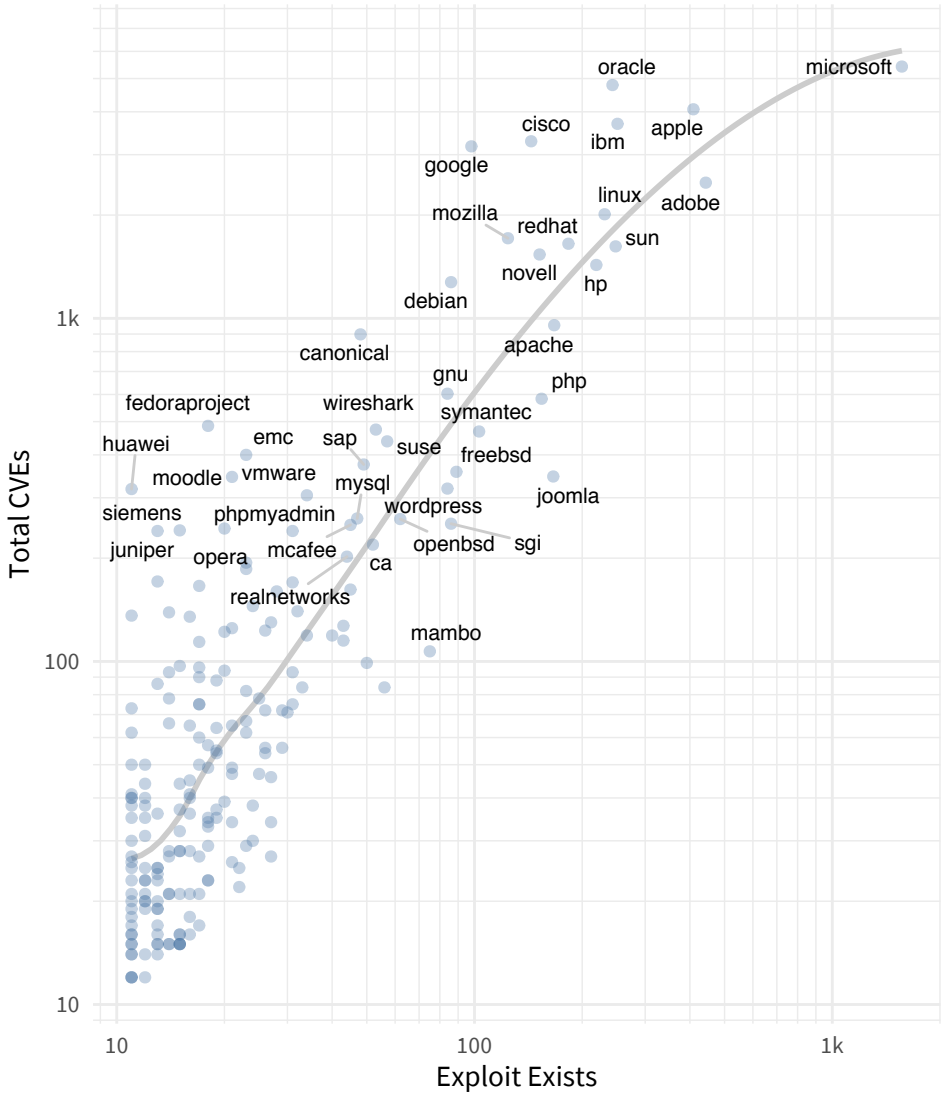
Source: Kenna / Cyentia

[3] *For more information on CVE Entry fields, see https://cve.MITRE.org/cve/identifiers/index.html*

Assuming an organization adopted such a strategy for remediation decisions, they could expect to achieve the results displayed in Table 1. Remediating all CVEs associated with the top 10 vendors (based on the total count of CVEs published for their products) leads to an efficiency score of 11.5% and 16.5% coverage (and you'd needlessly fix nearly 28,000 CVEs). To put that in perspective, you could expect double the efficiency and coverage by randomly choosing the same number of CVEs to remediate from the 94,000+ in this study. So a pair of dice would trounce a purely vendor-driven decision model. Clearly we need to expand our search for correlation to different attributes.

FIGURE 8

Vendors plotted according to the number of CVEs for which exploits exist (x) and total CVEs published (y)



Source: Kenna / Cyentia

CVSS Scores

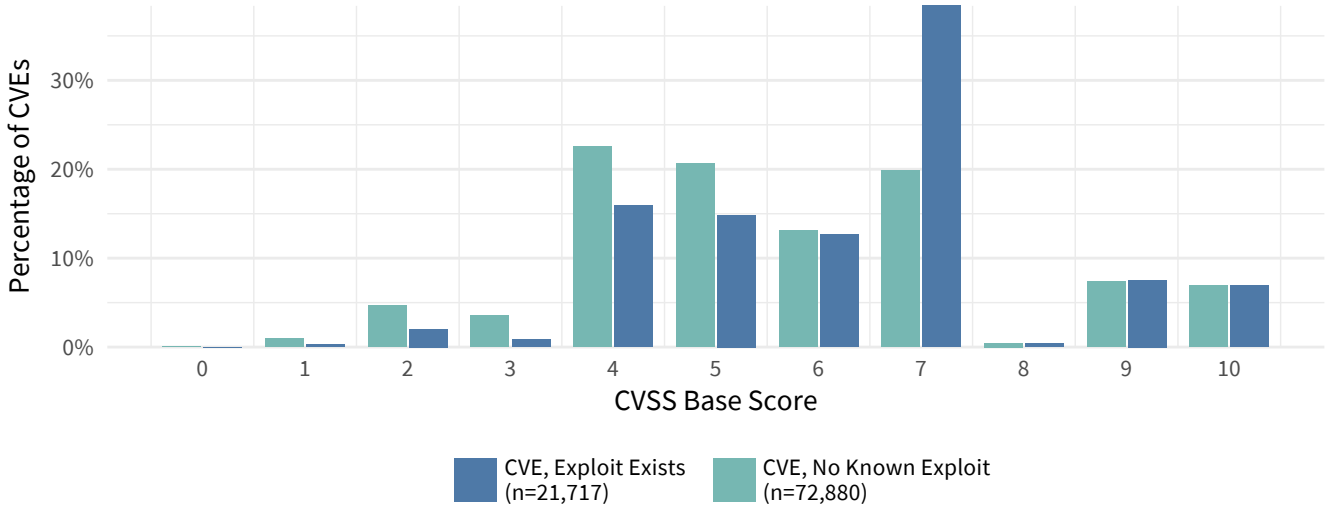
Another aspect of CVEs often considered during remediation decisions is the CVSS score, which is included for over 90% of all CVEs in the NVD. Based on a scale of 1 to 10, CVSS scores reflect assessments of the underlying vulnerability characteristics used to generate the scores. We will examine some of these in a moment, but first let's look at a plot of the overall scores in Figure 9.

[4] *CVSS calculator: https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator?vector=(AV:N/AC:L/Au:N/C:P/I:P/A:P)*

It is apparent that most CVEs fall in the 4 to 7 range for CVSS scores. Intuitively, one would expect to see increasing rates of exploitation among higher scores, and Figure 9 does support this notion to some extent. Only scores of 7 or more show a higher percentage of exploited CVEs than CVEs with no known exploit code or observations. That said, the ratio is near even for CVSS 6 and not too far off for CVEs that score a 4 or 5. At best, we see some evidence here of a positive (but not strong) correlation between CVSS scores and exploitation. Let's dig deeper.

FIGURE 9

CVSS Base Score by percentage of CVEs with existing exploits vs no know exploits

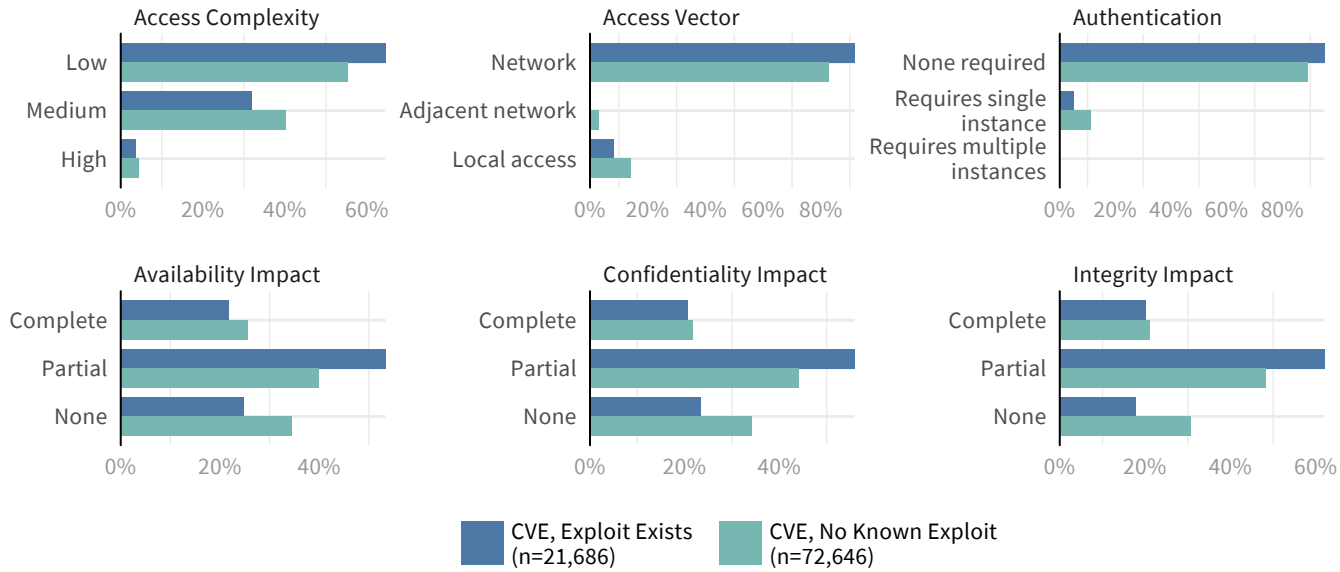


Source: Kenna / Cyentia

As mentioned previously, several underlying characteristics form the basis of a CVSS score. Figure 10 mimics the same basic format of Figure 9, except that it shows the primary components of CVSS rather than the overall score. CVEs exhibiting characteristics such as low access complexity, network vector, and no authentication do seem to attract more exploit-related attention, but they also feature prominently in a lot of non-exploited CVEs too. It is interesting to note that the tallest blue peaks (highest ratio of CVEs where exploits exist) yield a CVSS score[4] of 7—the tallest peak in Figure 9. Overall, nothing appears to stick out from Figure 8 as an unusually strong predictor of a CVE being exploited.

FIGURE 10

CVSS Base Score Metrics by percentage of CVEs with existing exploits vs no known exploits



Source: Kenna / Cyentia

As with vendor vulnerabilities, the big question is whether CVSS scores are useful for exploitation predictions and remediation decisions. There are actually several public examples of prioritization strategies based on CVSS scores. For example, a policy[5] from the Forum of Incident Response and Security Teams (FIRST) suggests patching everything with CVSS 7 and above. How would such a policy fare in light of historical CVE data?

TABLE 2

Results for prioritization strategies based on CVSS Base Scores

Remediate above CVSS Base Score	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	**23.1%**	**7%**	23%	7.1%
9	3,148	18,569	10,405	62,475	**23.2%**	**14.5%**	23%	14.7%
8	3,228	18,489	10,736	62,144	**23.1%**	**14.9%**	23%	15.1%
7	11,562	10,155	25,180	47,700	**31.5%**	**53.2%**	23%	39.8%
6	14,320	7,397	34,715	38,165	**29.2%**	**65.9%**	23%	53.2%
5	17,547	4,170	49,753	23,127	**26.1%**	**80.8%**	23%	73%

Source: Kenna / Cyentia

According to the test results in Table 2, a prioritization strategy of "remediate all CVEs with CVSS 7 or higher" would achieve coverage of 53.2% with an efficiency of 31.5%. This turns out to be the best balance among CVSS-based prioritization strategies, so FIRST's guidance appears sound from this perspective. Even so, this would still result in needlessly addressing over 25,000 CVEs. Strategies based on CVSS scores of 8 or more perform no better than chance, from an efficiency and coverage standpoint.

Reference Lists

A final aspect we'd like to explore concerns the list of references cited in most every CVE. The references contain one or more URLS listing out sources of information about the vulnerability. These may be the initial disclosure, discussions, any acknowledgement from the vendor and other public sources of information. Perhaps the appearance on certain lists focused on vulnerability reports and advisories indicates a higher likelihood of exploitation.

For example, "xf" refers to the IBM X-Force Exchange list, "bid" list is where vulnerabilities are listed with an official bugtraq ID and "confirm" reference is "when a vendor confirms an issue with its own advisory". Maybe the priorities, strengths and uniqueness of these references will give some (un)intended insight into the CVEs discussed and referenced. We focused on commonly-referenced lists for this analysis.

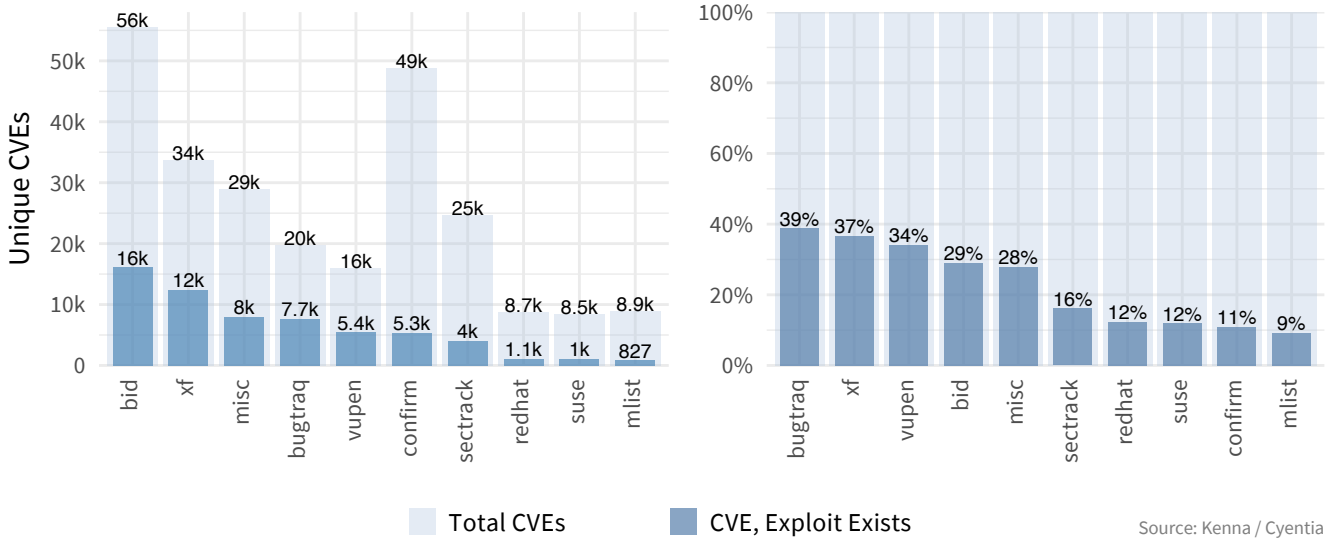
The left side of Figure 11 shows reference lists most often cited within CVE Entries, with focus on the number CVEs with existing exploits. The right side contains that same information, but presented as a percentage. The main takeaway is that these lists contain a signal and for whatever reason, some seem to be better indicators than others that a CVE will be exploited.

[5] *Example of CVSS based Patching Policy: https://www.first.org/cvss/cvss-based-patch-policy.pdf*

While we don't know of any particular strategy in practice that will use just the presence of a specific reference as the only deciding factor for remediation, we can appeal to the Table of Truth to at least explore, provide context and give us interesting "what if" insight into these references. Full Disclosure, Bugtraq, and BID perform better than random chance in terms of achieving good coverage and efficiency. But looking at Table 3 as a reflection of the references in general, suggests that the inclusion of reference lists as a factor in prioritization models is probably a good idea, but they are not going to get us where we want to be in isolation.

FIGURE 11

CVE References by volume and percentage with existing exploits



Source: Kenna / Cyentia

That last statement could be taken as the moral of this section. None of these attributes appear to be sufficient in and of themselves to optimally balance vulnerability remediation coverage and efficiency. But several of them do seem to offer some marginal utility over basing decisions on random chance, which appears promising. In the next section, we attempt to build a "Model of Everything" that leverages all available information to facilitate prioritization decisions.

TABLE 3

Results for prioritization strategies based on reference lists

Remediate from Lists	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	**16.1%**	**18.3%**	23%	26.7%
ms	1,054	20,663	2,714	70,166	**28%**	**4.9%**	23%	4.1%
fulldisc	1,300	20,417	2,308	70,572	**36%**	**6%**	23%	3.9%
confirm	5,296	16,421	43,480	29,400	**10.9%**	**24.4%**	23%	52.9%
bugtraq	7,695	14,022	12,139	60,741	**38.8%**	**35.4%**	23%	21.5%
bid	16,100	5,617	39,462	33,418	**29%**	**74.1%**	23%	60.2%

Source: Kenna / Cyentia

Exploit Prediction Model

We've established the need to prioritize remediation efforts and laid out the time and informational constraints on those decisions. We've also run through some basic rule-based prioritization strategies. But we haven't really brought up the word "prediction". Although, if you think about it, every decision to remediate is already a prediction about the future. Saying that one or more CVE's should be remediated is basically a prediction that they are likely to be exploited at some point in the future. The trick is, as we've seen, is that it's difficult to find a good strategy built on simple heuristics.

Even though the term "machine learning" (and "artificial intelligence") may be overhyped, oversold, misused, and abused in security, this type of prediction is a perfect problem for a classification algorithm. And we have a great data set to work with—over 21,000 vulnerabilities fall into the "exploit exists" category and each of them have dozens of features describing the vulnerabilities and providing context.

Classification Algorithms

We've already introduced the concepts of efficiency and coverage (which map directly to the traditional concepts of precision and recall). But what may not be apparent is that many classification algorithms don't just output a class. They can also output predictions across a range that reflects a confidence, or strength of the estimates. Predictions at the higher end of the range are much more likely (according to the model) to be exploited, while predictions at the lower end could probably be safely delayed for some period of time. If the model is doing its job, setting a higher threshold on the predictions will result in a high-efficiency model. And as remediation continues into the lower predictions, the coverage should approach 100%.

But this isn't a faith-based approach; we can evaluate how well different approaches work. We split our CVEs into two (random) groups, and use the first group to generate a model and the second group to test how well the model performs. Without overcomplicating our explanation, we're able to use this data-partitioning approach to select the best algorithm and tune our final model. Another thing we can do is to generate several models employing different sets of variables, and that's what we did. In order to see how different CVE efforts stack up, we trained up several different models:

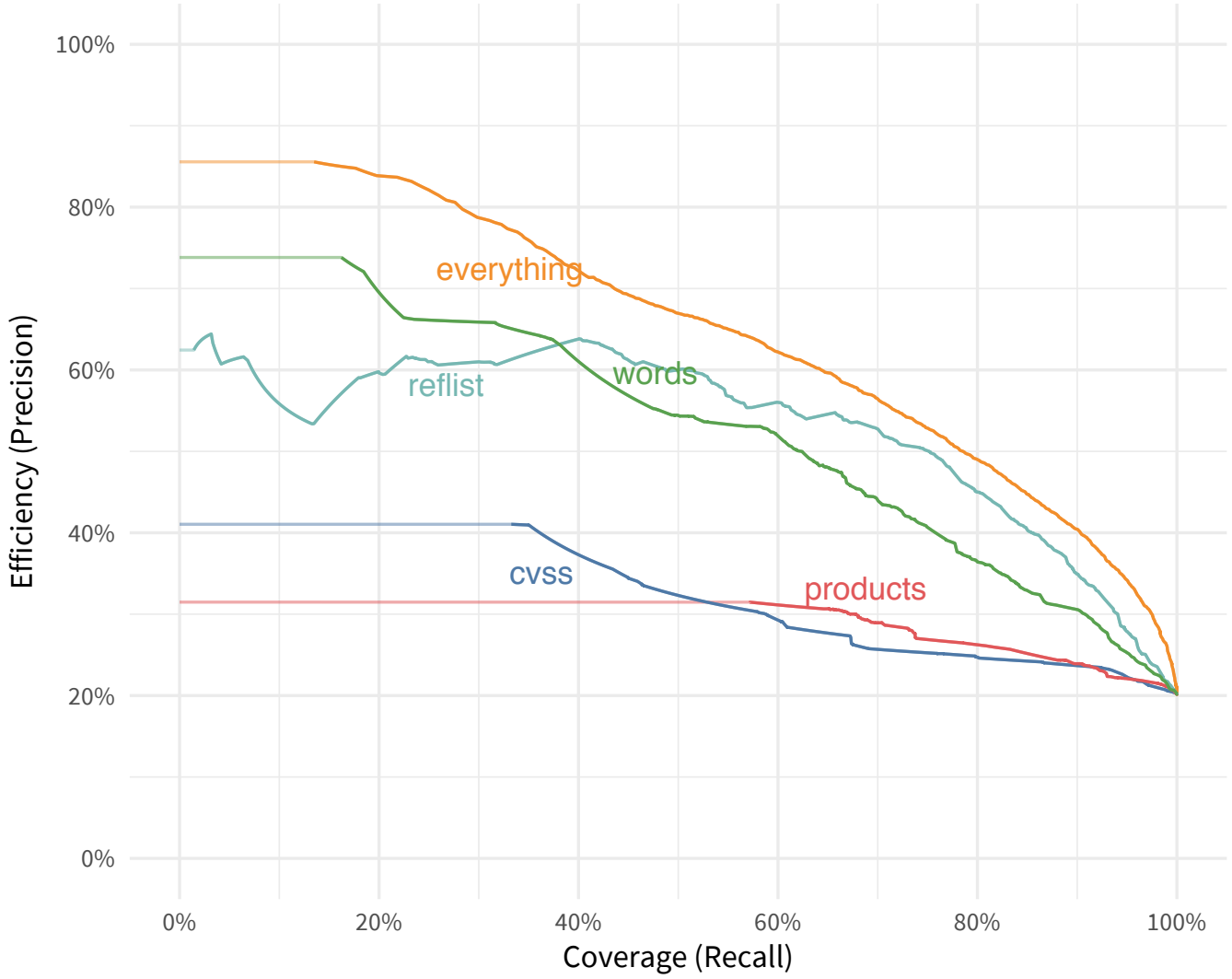
- **CVSS** is built with the CVSS base vectors and computed score.
- **Products** is built from CPE data on the majority (based on "popularity") of vendors and products.
- **RefLists** is built with each categorical reference listed with each CVE.
- **Words** is built with key words and phrases extracted from the free text description in each CVE.
- **Everything** incorporates all the above plus a few derived features that didn't fit into the above categories.

Keep in mind that these models will output a continuous variable. So, the lines in the next plot represents the trade-off between coverage and efficiency. As we slide down the range of outputs, the line slides down towards the remediation everything in the lower right side.

In our coverage/efficiency plots, we want to be as close to the upper right corner as possible, but each model will have its limit and we can only set a decision point somewhere along the line.

FIGURE 12

Coverage/Efficiency trade-offs for various predictive models.



Source: Kenna / Cyentia

Key Features of the Model

It's clear from looking at the above plot that the specific vendor and product as well as the CVSS vectors don't make good models on their own. Seeing the key words from the CVE description as well as the reference lists pushing above is certainly interesting. There's little doubt that the best model is leveraging all the variables.

The "Everything" model accounts for interactions among variables. This means it will compare multiple values together. For example, perhaps it's important that the description field contains the word "remote", or maybe it's helpful to know the CVE was discussed on bugtraq. But it could be even more important to know it contained "remote" and was discussed on bugtraq. That's part of the strength of moving beyond simple rule-based approaches.

One drawback of a complex model is that it isn't easy to determine exactly how each individual variable contributes to the overall model. But we can look at the relative contributions of variables and identify several key threads:

- **Remote Code Execution**: It sticks out across multiple features. Words in the summary such as "execute arbitrary code" and "remote attack" tend to help differentiate these CVE's.
- **Reference Lists**: It's logical that more severe vulnerabilities will have more discussions and some mailing lists and sources tend to be better indicators than others, yet overall these are good indicators of CVEs where exploits exist.
- **Vendors and Products**: While products on their own didn't create a good model, CPE values did contribute to the "Everything" model. The strongest value from CPE was simply the raw count of affected products.
- **CVSS Score**: We hoped CVSS v2 scores and/or vectors would be helpful in predicting exploits, but such was not the case. The leading contributor from CVSS (which was still a rather weak) was whether or not the availability impact was "partial".

A Model Finish

Since the model output covers a range, it's possible to set decision points along the line in the coverage/efficiency plot. For comparison, we selected three different prioritization strategies with the output from the "Everything" model. First, we created a "highly efficient" model. From a prioritization strategy, if an organization has limited resources, they could start here and have confidence that their investments would be targeting CVEs most likely to be exploited. Next, we created a "balanced" model, which is less efficient, but improves the overall coverage. Finally, we generated a "broad coverage" model that focuses on expanding coverage at the cost of efficiency.

TABLE 4

Results for prioritization strategies based on varying thresholds for prediction model[6]

Remediation Model	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
Highly Efficient	5,546	13,518	1,450	74,083	**79.3%**	**29.1%**	23%	16.7%
Balanced	11,755	7,309	7,399	68,134	**61.4%**	**61.7%**	23%	45.8%
Broad Coverage	15,550	3,514	16,917	58,616	**47.9%**	**81.6%**	23%	77.6%

Source: Kenna / Cyentia

Now we are able to compare all remediation strategies outlined in this document on the same chart based on the level of coverage and efficiency achieved by each one. We do this in Figure 13, where points on the plot represent the strategies and the size of those points corresponds to the total number of CVEs remediated by that strategy.

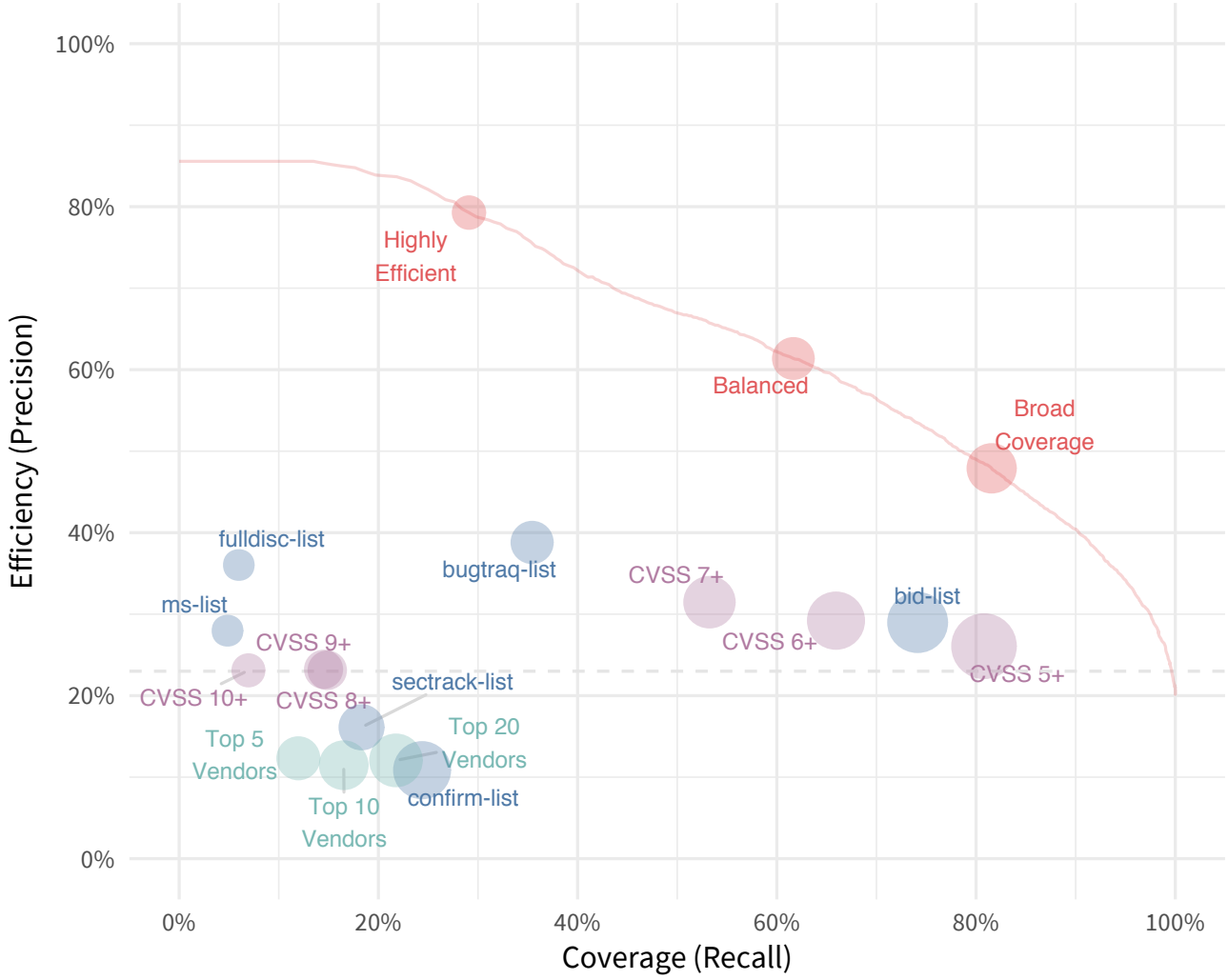
Take a moment to orient yourself in Figure 13, and examine where the different strategies fall with respect to efficiency and coverage. As you take it all in, notice that all of the rule-based strategies fall below the midway point on the efficiency axis. It seems safe to assume that any vulnerability management program relying on such strategies for remediation decisions will inevitably expend a considerable amount of resources on issues that don't truly matter.

Also apparent is the tension between coverage and efficiency. Remediating every CVE with a CVSS 5 or above, for example, will yield an impressive 81% coverage but a not-so-impressive 26% efficiency. Furthermore, Table 2 informs us that we should expect to achieve 73% coverage by randomly picking out CVEs to remediate. This strategy is better than random chance, but not by much and it involves a large number of false positives.

[6] *These numbers are derived from only half the data (the "holdout" or "test" data), and we used that output to infer how it would do on all the CVEs. This enables comparison to the other models since the sample sizes are the same.*

FIGURE 13

Coverage/Efficiency plot for various prioritization strategies and prediction model thresholds



Source: Kenna / Cyentia

It is rather obvious from Figure 13 that the "Everything" model we developed outperforms all other strategies, regardless of whether your goal is to maximize coverage, efficiency, or balance both. When compared to a strategy of remediating all CVEs with a CVSS score of 7 or more, our model achieves twice the efficiency (61% vs. 31%), half the effort (19K vs. 37K CVEs), and one-third the false positives (7K vs. 25K CVEs) at a better level of coverage (62% vs. 53%). It performs 8X more efficiently than a strategy based on remediating vulnerabilities from the top 20 vendors from Figure 8. It does this, in short, by incorporating multiple different types of data points and accounting for interaction effects.

Something else to note is that rule-based models don't offer much in the way of differentiation. With the predictive model, numerous variables (over 250 features used for this research) output a continuous range, which allows for subtle and meaningful differentiation. Starting at the top of the range and working your way down the prediction line will guarantee a more efficient and flexible process that is better able to handle the volume and velocity of new vulnerabilities.

Furthermore, the results presented above represent a worst case scenario for decision making. In reality, nobody considers ALL published CVEs for potential remediation. Any vulnerability program worth its salt will take steps to reduce that funnel to a more manageable level. For instance, only those products present in the environment (an/or deemed critical) will be "on the table" for remediation decisions. The point is that the model proposed here will make existing strategies even better and vice versa. The key lesson is that the "best" blended strategy will become apparent through the measures of efficiency and coverage.

Conclusion

In an article published in The Atlantic, security luminary Bruce Schneier asked the question: "Are vulnerabilities in software dense or sparse?" Many security practitioners know the answer to that question all too well. We experience the daily deluge of new disclosures. We see it in the backlog of forgotten vulnerabilities found by our scanners. In light of such density, our frantic efforts to remediate vulnerabilities may have little to no effect on risk. And that is disheartening…but there is hope.

That hope lies in changing the way we think about vulnerabilities, and vulnerability management. Many of our problems come from seeking the wrong goal. The question was never "how many vulnerabilities?" It was always "how much risk?"

This report attempts to answer a more specific question about how we measure and address these risks, namely:

- What current strategies reduce the most risk?
- Can predictive models outperform those strategies?

Tradeoffs always exist – efficiency trades off with coverage even in the best models. We as a security industry must develop risk reduction strategies that are most effective for the individual organization. But if we want to advance the state of the industry, we have to play with variables that we can affect:

1. WE NEED TO SPEED UP. Once a vulnerability is published it typically takes just two weeks for corresponding exploit code to get published (and the overwhelming majority of exploits in the wild are based on known exploit code). That means that we need automated technologies designed to help organizations rapidly understand which vulnerabilities pose the greatest risk within the context of their businesses.

2. WE NEED TO PRIORITIZE. Success and failure of enterprise security teams and its leaders should be measured, tracked and adjusted based on metrics that can accurately quantify risk and the impact of efforts to reduce them, not a simple number of vulnerabilities closed. Comprehensive and unified metrics enable enterprises to understand the effectiveness of remediation strategies based on overall impact instead of amount of effort.

3. WE NEED TO PREDICT. Technology is a critical enabler (and oftentimes headache) of success in today's digital economy. Leveraging technologies like machine learning backed by strong data science enables us to build predictive models that can be measured against, learned from and improved upon. These predictive models are critical to proactively reduce risk efficiently and effectively.

Back inside the enterprise, our advice is to start small: measure the efficiency and coverage of your remediation strategy. See what policy tweaks increase those metrics and begin the OODA loop that we've walked through in this report. From there, the value of predictive models will become not just apparent, but measurable.

Predictive models, like those developed in this report, can and do enable businesses to adopt a proactive strategy for vulnerability remediation that delivers the most efficient use of their people, tools, time, and ultimately dollars to address the threats that pose the greatest risk.

This report is a first pass at building a predictive model from a large sample of live asset and vulnerability data as well as in the wild attacker behavior. As the underlying data evolves, we'll keep you posted on the trends, but more importantly on strategies that the data indicates are most effective.

PRIORITIZATION TO PREDICTION

CYENTIA INSTITUTE

> "When compared to a strategy of remediating all CVEs with a CVSS score of 7 or more, our model achieves twice the efficiency (61% vs. 31%), half the effort (19K vs. 37K CVEs), and one-third the false positives (7K vs. 25K CVEs) at a better level of coverage (62% vs. 53%)."