KENNA Security

Cyentia 119 INSTITUTE

PRIORITIZATION TO PREDICTION

Volume 7: Establishing Defender Advantage

KENNA Security

PRIORITIZATION TO PREDICTION

VOLUME 7: ESTABLISHING DEFENDER ADVANTAGE

This research was commissioned by Kenna Security. Kenna collected and provided the dataset to the Cyentia Institute for independent analysis for this report.

Kenna Security is the enterprise leader in risk-based vulnerability management. Kenna Security solutions enable organizations to work cross-functionally to determine and remediate cyber risks. They leverage machine learning and data science to track and predict real-world exploitations, empowering security teams to focus on what matters most. Headquartered in Santa Clara, Kenna serves nearly every major vertical and counts CVS, KPMG, and many Fortune 100 companies among its customers.

Find out more at www.kennasecurity.com.

Cyentia INSTITUTE

Analysis for this report was provided by the Cyentia Institute. Cyentia is a research and data science firm working to advance cybersecurity knowledge and practice. We do this by partnering with security vendors and other organizations to publish a range of high-quality, data-driven content like this study.

Find out more at www.cyentia.com.

Overview & Key Findings

"I have steadily endeavoured to keep my mind free so as to give up any hypothesis, however much beloved, as soon as facts are shown to be opposed to it."

–Charles Darwin

Welcome to lucky #7 in the Prioritization to Prediction (P2P) report series. If you're a P2P veteran, welcome back. If you're a rookie, thanks for joining us! We have good stuff in store for both groups.

The Prioritization to Prediction series was created to improve the practice of risk-based vulnerability management (RBVM) by analyzing factors that drive vulnerability remediation and exploitation. More so than any prior volume, this one's focused on answering some very specific—and very important—questions left hanging out there from the last report. The two biggies are:

1. Does releasing exploits before patches are available help or harm defenders?
2. Which software products offer the greatest degree of advantage to defenders?

You probably have opinions on those questions, but, speaking frankly, we're not interested in opinions for this report—including our own. We're interested in gathering hard evidence, formally testing hypotheses, and reaching well-justified conclusions. In other words, we plan to approach this admittedly touchy subject scientifically. If that's not what you're looking for, perhaps your time would be better spent scrolling through the comments section on any of the numerous posts covering this topic. But if you want objective data analysis, this is the right place. Let's get to it.

Key Findings

Relatively few vulnerabilities are exploited.

Exploit code exists for 6.5% of vulnerabilities published since 2017, and 3.7% of them record active exploitation in the wild (1.3% show both). Over time, the barrier between proof of concepts and active attempts to exploit enterprise assets appears to be diminishing.

Releasing exploits before patches harms defenders.

The timeline of exploitation shifts an average of 98 days earlier when exploit code emerges before patches are released. We ruled out all other hypotheses that might explain this shift. Irresponsible exposure has a cost.

Exploit code leads to faster and wider exploitation.

Among vulnerabilities exploited in the wild, those with exploit code (about ⅓) mount up nearly 15X the overall exploitation activity (across 6X as many organizations) of those without exploit code!

Attacker-defender advantage varies widely across software products.

It takes over 40X longer for organizations to fix vulns in Linux and SAP software (~900 days) than to reach that same milestone with Google and Microsoft products (~22 days). Remediation always outpaced exploitation for those latter two, whereas Siemens and Wordpress held defender advantage for just 3/24 months.

In Our Last Episode...

In Prediction to Prioritization Volume 6, we had the rare opportunity to measure the effect of vulnerability disclosures and exploit development on attacker-defender dynamics. To do that, we analyzed key milestones in the lifecycle of numerous vulnerabilities, remediation efforts across hundreds of organizations, and large-scale detections of exploitation activity. Putting all that together allowed us to produce the culminating chart in that report, which we've included for your convenience below.

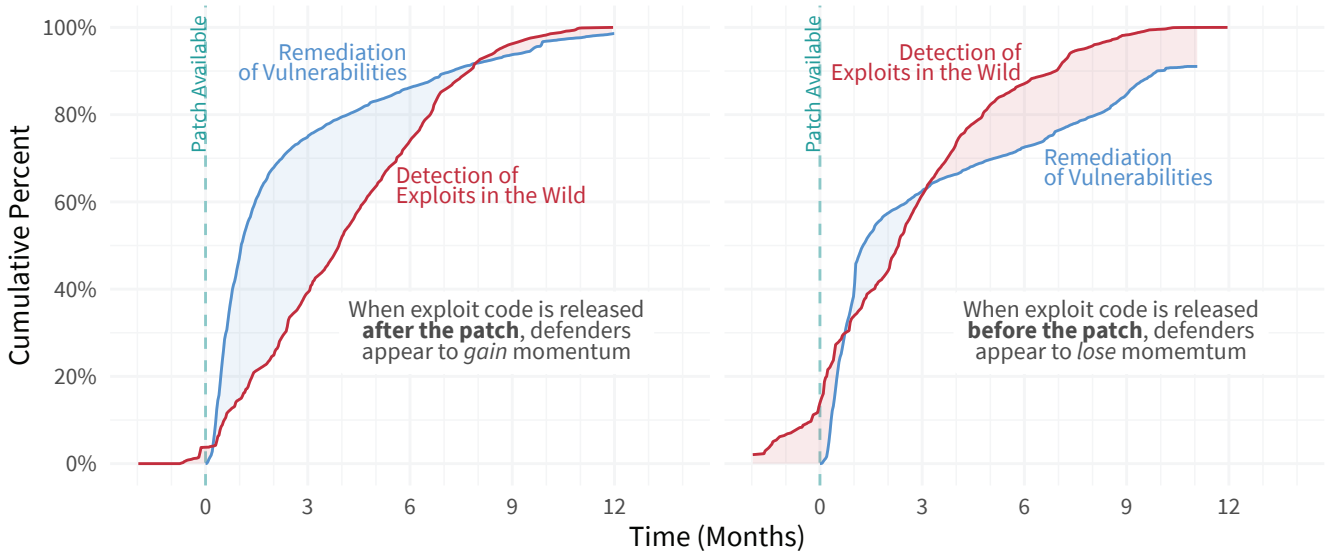

Figure 1: Comparison of attacker-defender divide when exploit code releases after vs. before patch availability

Let's focus on the left side of Figure 1 for now. The blue line shows the average vulnerability remediation timeline across several hundred organizations. The red shows the spread of exploitation from the first organization that detected related activity to the last. The key takeaway is that defenders control the momentum (shaded blue area) most of the time—assuming the patch becomes available before exploit code is published.

The right side of Figure 1 proves that the attacker-defender divide is not set in stone. One minor change—exploits releasing before patches are available—impacts the exploitation and remediation timelines. The difference between the two scenarios in Figure 1 is unmistakable. The period of defender advantage is largely erased when exploits drop before patches, largely because the exploitation timeline shifts an average 47 days earlier.

Not wanting to jump to conclusions, we offered the following hypotheses for this apparent shift:

1. Hypothesis #1: Releasing exploit code early leads to earlier remediation, thus helping defenders.
2. Hypothesis #2: Releasing exploit code early leads to earlier detection, thus helping defenders.
3. Hypothesis #3: Releasing exploit code early facilitates earlier exploitation, thus helping attackers.

Examining the Evidence for Early Exploits

With that background, we're all set to review evidence collected since the last report to test these hypotheses. Ain't it fun when you get to do science in cybersecurity? We'll address each hypothesis in turn, but first let's go over the sources and scope of the evidence we're using.

Remediation and Exploitation Data

Kenna provided Cyentia with an anonymized extract of vulnerability scan and remediation data from nearly 500 organizations to analyze for this report. All told, that dataset comprises over 6 billion vulnerabilities affecting 13 million active assets under management in the Kenna Security platform. This represents a sample of organizations meeting certain conditions allowing for meaningful analysis, such as minimum thresholds for the number of vulnerabilities, assets in scope, level of activity, length of time using the platform, etc.

Kenna also contributed data on exploit code availability and exploitation in the wild via a range of intelligence sources they incorporate to help identify high-risk vulnerabilities. We also include exploitation data, once again generously shared by Fortinet. Fortinet has one of the largest security device footprints in the industry,† granting wide visibility into exploit detection. They continue to support exploitation research like this and the Exploit Prediction Scoring System (EPSS). Our thanks to Fortinet and others who give back to the security community in this way.

For this analysis, we focus on vulnerabilities published to MITRE's Common Vulnerabilities and Exposures (CVE) List since 2017. That includes almost 60,000 CVEs, about half of which were detected via vuln scanners by organizations in this study. Exploit code exists for 6.5% of these CVEs, and 3.7% of them record active exploitation in the wild (1.3% show both). The chart to the right in Figure 2 packages all those CVE stats into a nice and neat Venn diagram.

A quick word on non-CVE vulnerabilities

It's true that focusing on CVEs misses vulnerabilities that, for one reason or another, fall outside the CVE process. We've clearly stated that limitation from the very first volume in this series. But how much of a limitation is that, actually? According to our data, about 95% of all observed vulnerabilities in our dataset have a CVE. We infer from these results that CVEs offer good coverage for the vulnerabilities identified by the various scanners used by organizations around the world as a foundation of their VM programs.

† *Source: IDC Worldwide Security Appliance Tracker, April 2020 (based on annual unit shipments of Firewall, UTM, and VPN appliances)*

We showed a similar breakdown of CVEs way back in P2P Volume 1 (recreated here on the left side), but the numbers were quite different. Most notable is a much higher proportion of exploit code and lower rate of active exploitation in the older dataset.

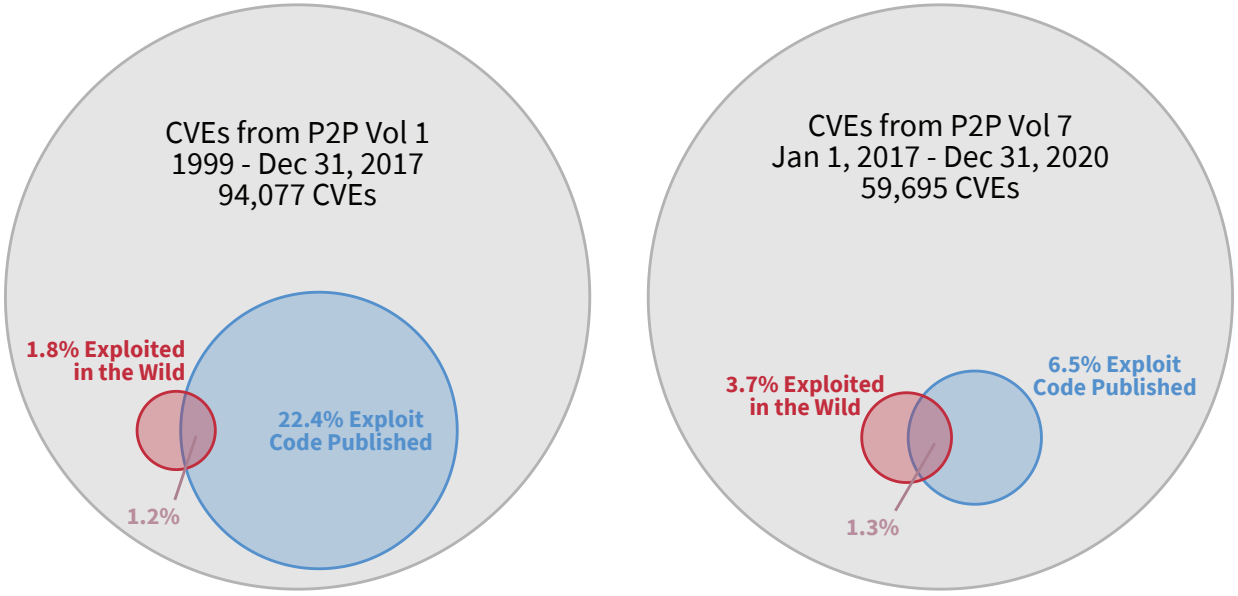

Figure 2: Proportion of published CVEs with exploit code and/or observed exploits in the wild

Several things account for the differences shown in Figure 2. First, P2P Volume 1 looked at the entire 20+ year history of the CVE List. Many of those vulns were exploited long ago. Second, 2017 (the start of Volume 7 data) greatly expanded the CVE Numbering Authorities (CNAs) program, resulting in a huge increase in CVEs published. Thankfully, exploit development hasn't grown at an equal scale. And finally, we've added additional sources for detecting exploitation in the wild, which ups that statistic.

Overall, Figure 2 conveys an important truth VM programs need to hear: the vast majority of vulnerabilities are not exploited. That doesn't mean they shouldn't be addressed. But it does mean that most vulns don't represent an immediate risk and can be safely deferred until the exploited ones are remediated. The challenge, of course, is figuring out what's what in an ever-deepening sea of published vulnerabilities. Especially since, as we'll soon demonstrate, the release of exploit code often foreshadows broad exploitation in the wild. That's why vulnerability management is so hard and why vulnerability intelligence is so important.

Exploit code vs Exploitation in the wild

We throw these terms around a lot in this report, so let's make sure we're all on the same page regarding what they mean. Exploit code includes proof-of-concepts or working ("weaponized") code for exploiting a vulnerability. Exploitation in the wild or active exploitation refers to exploit activity targeting vulnerabilities detected by host and network-based security sensors, reverse engineering malware, etc.

Testing the Hypotheses

> "The great tragedy of science - the slaying of a beautiful hypothesis by an ugly fact."
>
> - Thomas Huxley

We've already referenced the finding from P2P Volume 6 that moves up the exploitation timetable by 47 days when exploits predate patches. Since we have more data with a longer history of vulnerabilities for this volume, it's worth checking that comparison again as we begin weighing the evidence on what's driving this shift. Figure 3 revises that early exploit shift to an average of 98 days (median = 90), more than doubling the previous result.

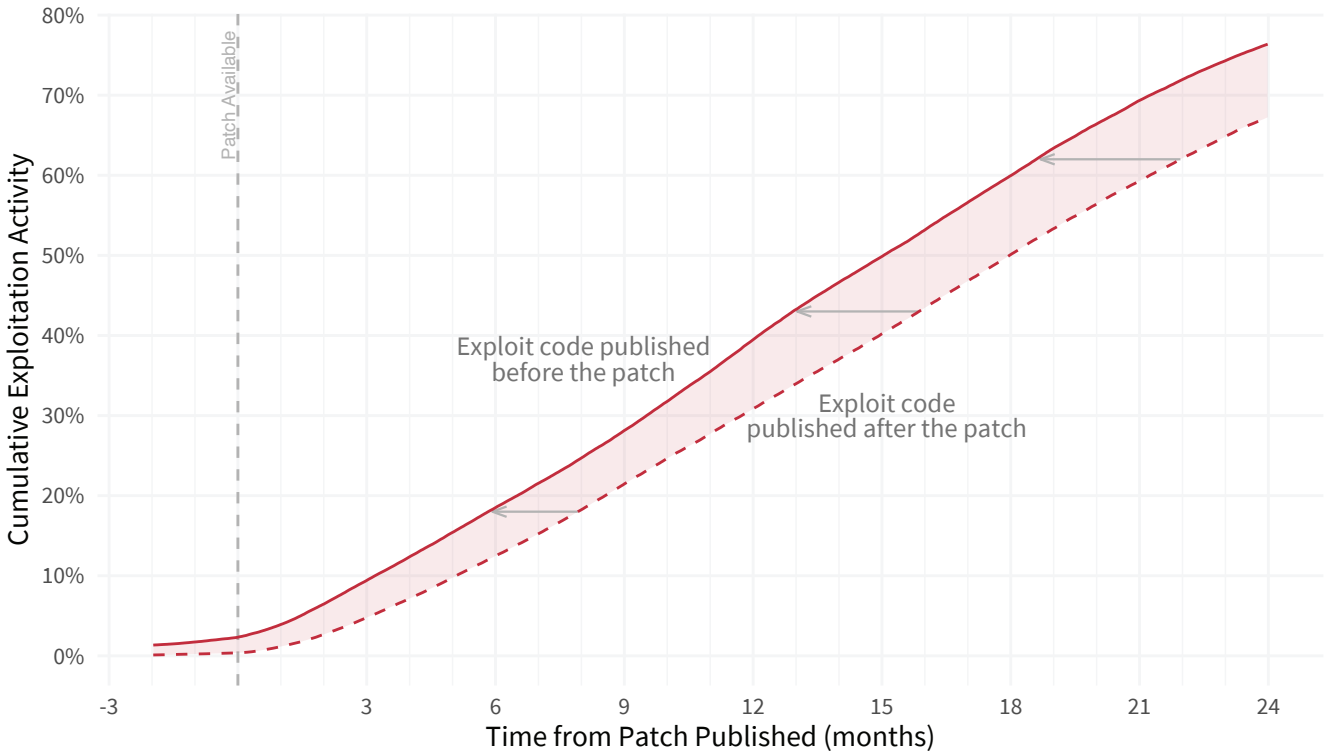

Figure 3: Exploitation timelines for vulnerabilities exploited before vs. after patch release

Don't get overly caught up with the 47 vs. 98 days, though. As seen in Figure 3, the exploit-first and patch-first lines diverge over time. So, the longer the span of time we examine, the higher the average shift will become. Let's just accept that a) publishing exploit code before patches moves up the exploitation timetable substantially and b) there are several plausible reasons for this shift. Speaking of...let's get to those hypotheses.

The timetable of exploitation in the wild moves up by an average 98 days when exploits predate patches. That shift continues to widen over time.

Hypothesis #1: Releasing exploit code early leads to earlier remediation, thus helping defenders.

Some suggest that releasing exploit code can be used to prove to vendors or asset owners that a vulnerability is real and/or enable admins to remediate prior to a patch. That's a logical suggestion, but we don't have to appeal to logic for this one. We have multiple ways of testing this hypothesis. Let's start by using survival analysis to compare remediation velocity between vulnerabilities where exploit code was released before vs. after the patch.

Remediation Velocity measures the speed and progress of remediation. How quickly are issues addressed and how long do they persist within and/or across assets?

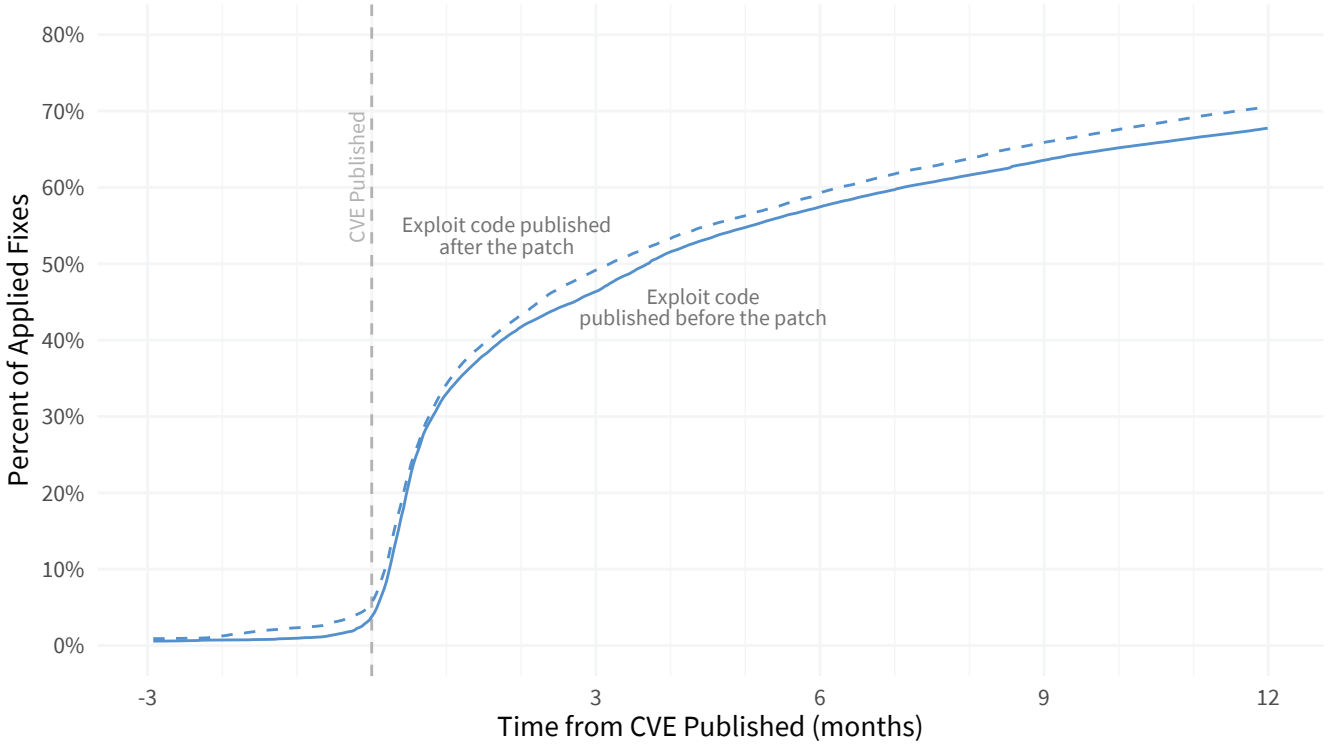

Figure 4: Remediation timelines for vulnerabilities exploited before vs. after patch release

Simply eyeballing Figure 4 is probably enough to betray how this test will pan out. There's just not much difference between these scenarios, and if forced to choose, we'd have to side with the alternative hypothesis—remediation appears faster when vulns are patched before they're exploited. But let's go one step further and put some actual numbers around this test.

There's not much difference in remediation velocity between vulnerabilities exploited before vs. after patch release. Thus, the argument that publishing exploit code early promotes quicker patch release holds little water.

Reviewing Key Remediation Metrics

4. **Area Under the Curve (AUC)** measures the area under the survival curve representing "live" (open) vulnerabilities. A lower AUC means higher velocity.
5. **Mean Time to Remediation (MTTR)** equals the average amount of time it takes to close vulnerabilities.
6. **Vulnerability half-life** represents the time required to close exactly 50% of open vulnerabilities.

Table 1 offers no fewer than three different metrics that all agree with what we see in Figure 4. Let's go with MTTR† because it seems the most intuitive and memorable in this context. In the first year after publication, vulnerabilities exploited prior to being patched are remediated an average of 10 days more slowly than those in the patched-first category.

	Exploit First	Patch First
1yr AUC	0.466	0.446 (faster)
MTTR	63 days	53 days (faster)
Half-Life	112 days	97 days (faster)

Table 1: Remediation velocity metrics for vulnerabilities exploited before vs. after patch release

After reviewing the evidence, we reject this hypothesis and conclude that releasing exploit code early **does not** lead to earlier remediation. Furthermore, we see no reason to believe this practice helps defenders. The data strongly supports the opposite is true—dropping exploits before patches are available **harms rather than helps** defenders.

Hypothesis #2: Releasing exploit code early leads to earlier detection of exploitation in the wild, thus helping defenders.

We've established that exploitation in the wild moves faster when exploit code predates patches. But it's possible that organizations just *detect* exploitation in the wild sooner because the existence of exploit code enabled quicker creation and distribution of IPS and AV signatures.

To test this hypothesis, we need to incorporate signature creation dates into our timeline and compare those to the earliest detections in the wild. A spike in detections immediately after deploying signatures would suggest that exploitation was occurring all along and that earlier detection benefits defenders. A lag between signature release and initial detections would suggest the opposite.

That means we need to introduce new data to make this comparison. We presented a bunch of milestones in the vulnerability lifecycle in P2P Volume 6, and we're not going to reiterate them all here. But take our word for it that the creation date of detection signatures was not among them. We've since remedied that omission and are now ready to test this hypothesis.

Let's start with the left chart in Figure 5, where we plot the spread of exploitation in the wild based on the first reported detection by any organization. There we see that 17% of CVEs show exploitation activity on the first day of signature deployment, suggesting that accelerating signature deployment could enable speedier detection of up to 17% of vulnerabilities. What this does not answer is whether releasing exploits early promotes faster signature deployment. Hold that thought for a moment.

† *We're using an approximation for mean-time-to-remediation for this because these curves don't close in the timeline analyzed.*

It's important to note that the left side of Figure 5 only looks at one date for every CVE—the date when it was first detected by an organization. It does not look beyond that "Patient Zero" to trace the spread of exploitation across the whole population of organizations. That's what the right side of Figure 5 does, and it tells a different story.

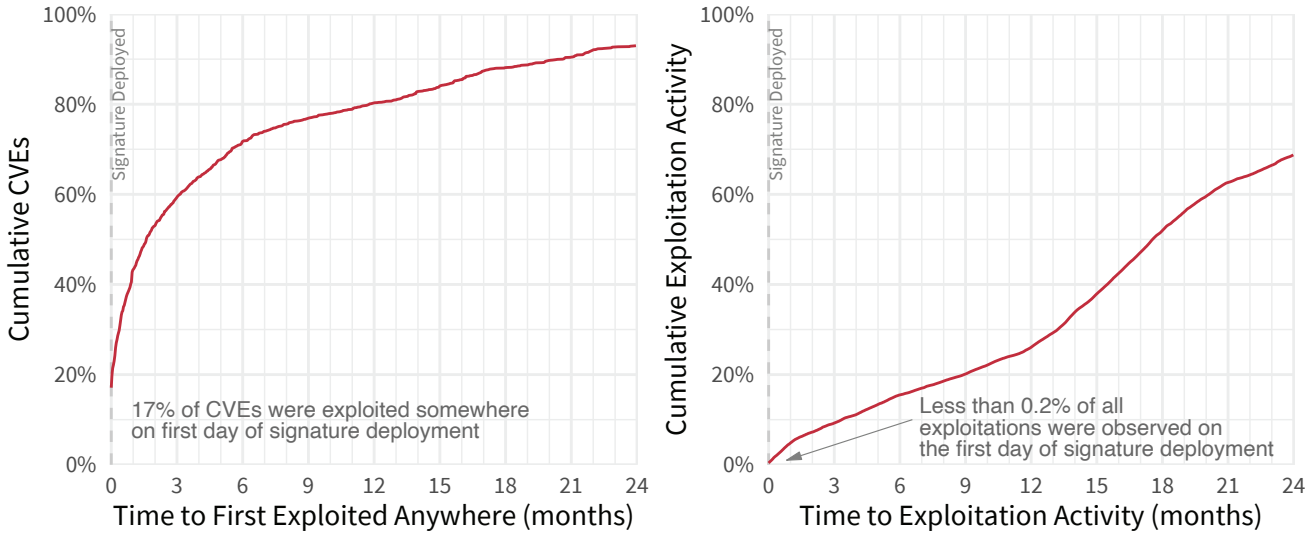

Figure 5: Spread of exploitation in the wild relative to release of detection signatures

The chart on the right in Figure 5 offers a more realistic view by considering all exploitation activity detected during our study's time frame. It shows a much more gradual spread of exploitations in the wild and pegs the day 1 rate at less than 0.2%. In other words, 0.2% of all known detections were recorded on the first day of signature deployment. That fact effectively lowers the potential scope of "helping defenders" by releasing exploits as soon as possible.

The right side of Figure 5 reveals another important truth: exploitation in the wild is more of a slow burn than an explosion. The detection of exploitation in the wild is often used to justify a no-holds-barred approach to releasing vulnerability and exploit information, regardless of whether patches are available. Figure 5 urges a much calmer and more coordinated approach to getting defenders the information they need without adding fuel to the exploitation fire.

Now let's circle back to the question of whether releasing exploits early promotes faster signature deployment. To test that, Figure 6 measures the number of days between CVE publication and availability of exploit detection signatures. Here we see the opposite effect—detection signatures emerge more quickly when patches come before exploits. We suspect this reflects a more coordinated disclosure process. Dropping exploits first, on the other hand, potentially disrupts that coordination and appears to slow production of detection signatures.

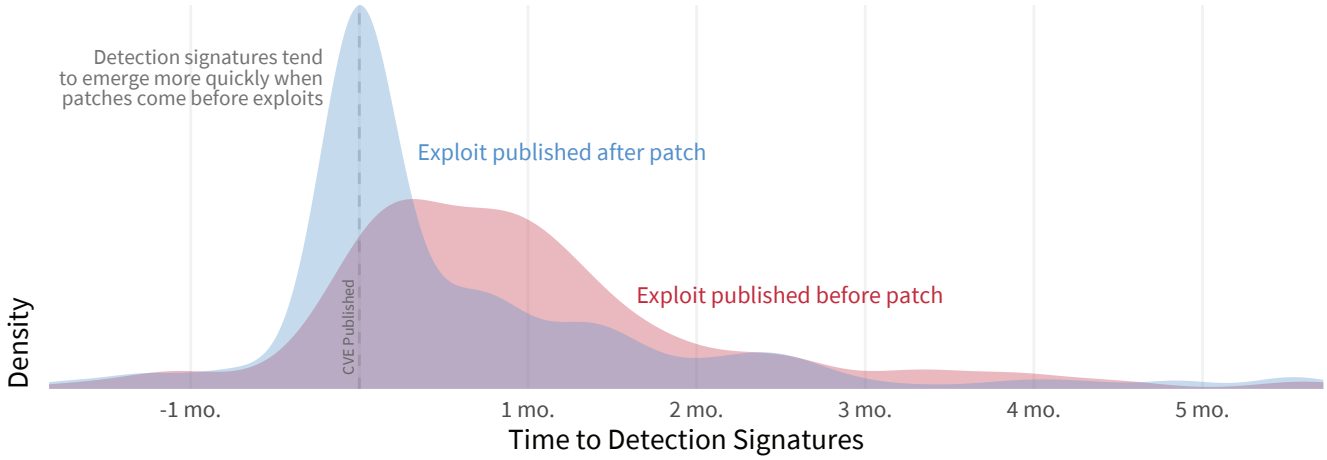

Figure 6: Timing of detection signature distribution for vulnerabilities exploited before vs. after patch release

Thus, while we do see some indication that deploying signatures earlier might aid earlier detection for a very limited minority of exploitations, the whole of the evidence does not support pre-patch exploit releases as a means of accomplishing this goal. Furthermore, data on signature timelines disputes the very premise of that argument: pre-patch exploits result in slower signature release. We therefore reject Hypothesis #2 and conclude that releasing exploit code before patches does not help defenders by speeding up signature deployment.

Hypothesis #3: Releasing exploit code early facilitates earlier exploitation, thus helping attackers.

The opening quotation from Darwin reminds us that science requires a willingness to give up a much-beloved hypothesis. It doesn't seem like a stretch to infer that it also implies being open to much-loathed hypotheses. We suspect this third hypothesis falls in the latter category for many because it goes against some strong presuppositions. But let's approach this with cool heads and hard data.

The outcomes of testing Hypotheses #1 and #2 are big parts of the requirement to test this hypothesis. Releasing exploits without regard to patch availability does not appear to help defenders, but that doesn't mean doing so helps attackers. Let's see what the data says specific to that aspect of vulnerability exploitation.

We've already seen data that convincingly argues releasing exploit code before patches become available increases the speed of exploitation in the wild. Now we want to know how exploit code affects the spread of exploitation activity.

Figure 7 compares exploitation prevalence for vulnerabilities with published exploit code (red) with those with no known exploit code (blue). Prevalence here is based on the proportion of all organizations that detected exploitation activity. The red density plot shifts noticeably right and has a fatter tail (even though the plot uses a log scale, which dampens that effect). Looking into the numbers behind the chart, we find vulnerabilities with exploit code target 6X the number of organizations compared to vulnerabilities without exploit code. It's hard not to interpret this as evidence that releasing exploit code ultimately puts more organizations at risk. We understand this inference will raise some hackles, but let's push through the discomfort to see what else the data has to say on this topic.

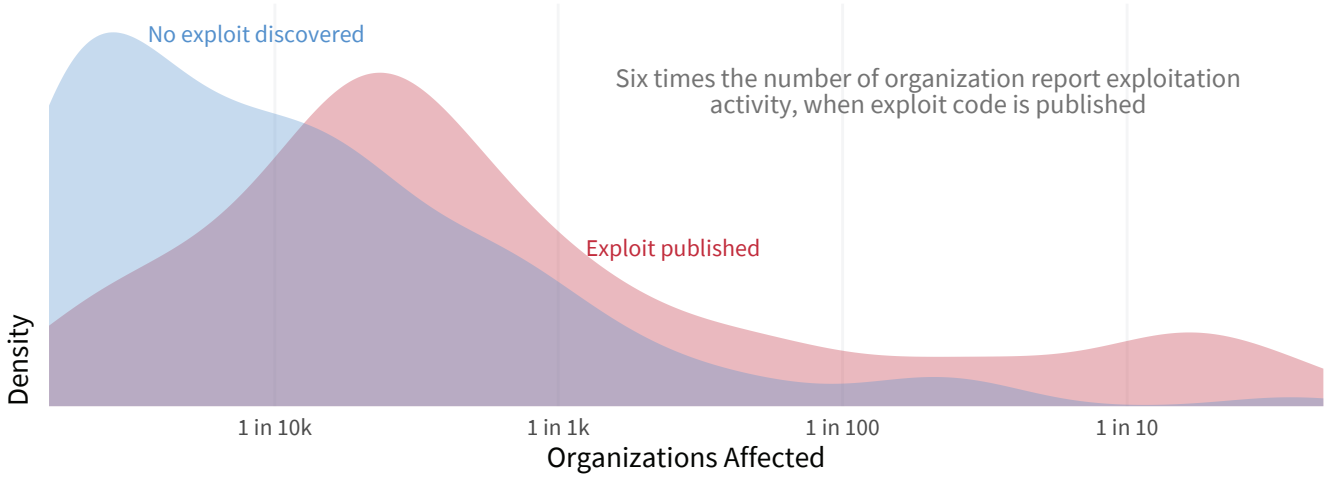

Figure 7: Prevalence of exploitation in the wild for vulnerabilities with vs. without published exploit code

Releasing exploit code ultimately puts more organizations at risk. We understand that this inference will raise some hackles, but let's push through the discomfort to see what else the data has to say on this topic.

Think of Figure 8 as a sort of marriage of Figures 5 and 7, combining the speed and spread of exploitation in the wild. Like Figure 7, it compares vulnerabilities with and without exploit code. And once again, the difference is pretty striking. Among vulnerabilities exploited in the wild, those with exploit code (about ⅓) mount up nearly 15X the overall exploitation activity (across 6X as many organizations, per Figure 7) as those without exploit code!

That's a big finding, but not entirely surprising. The whole point of releasing exploit code is to demonstrate how a vulnerability might be exploited or to actually do so (ethically or otherwise). Once out there, that code is incorporated into (malicious and legit) hacking tools, weaponized by malware authors, etc. This effectively makes it easier and cheaper to find and exploit related vulnerabilities at scale.

Since many vulnerabilities can be exploited without exploit code (e.g., SQL injection), it could be argued that the blue 'No exploit code found' line in Figure 8 is inflated. As a way of leveling the playing field for that comparison, we filtered the data to just vulnerabilities that allow remote code execution (RCE). The results are even more dramatic. RCE vulnerabilities with published exploit code registered a nearly 30X increase in overall exploitation activity compared to RCE vulns with no exploit code available!

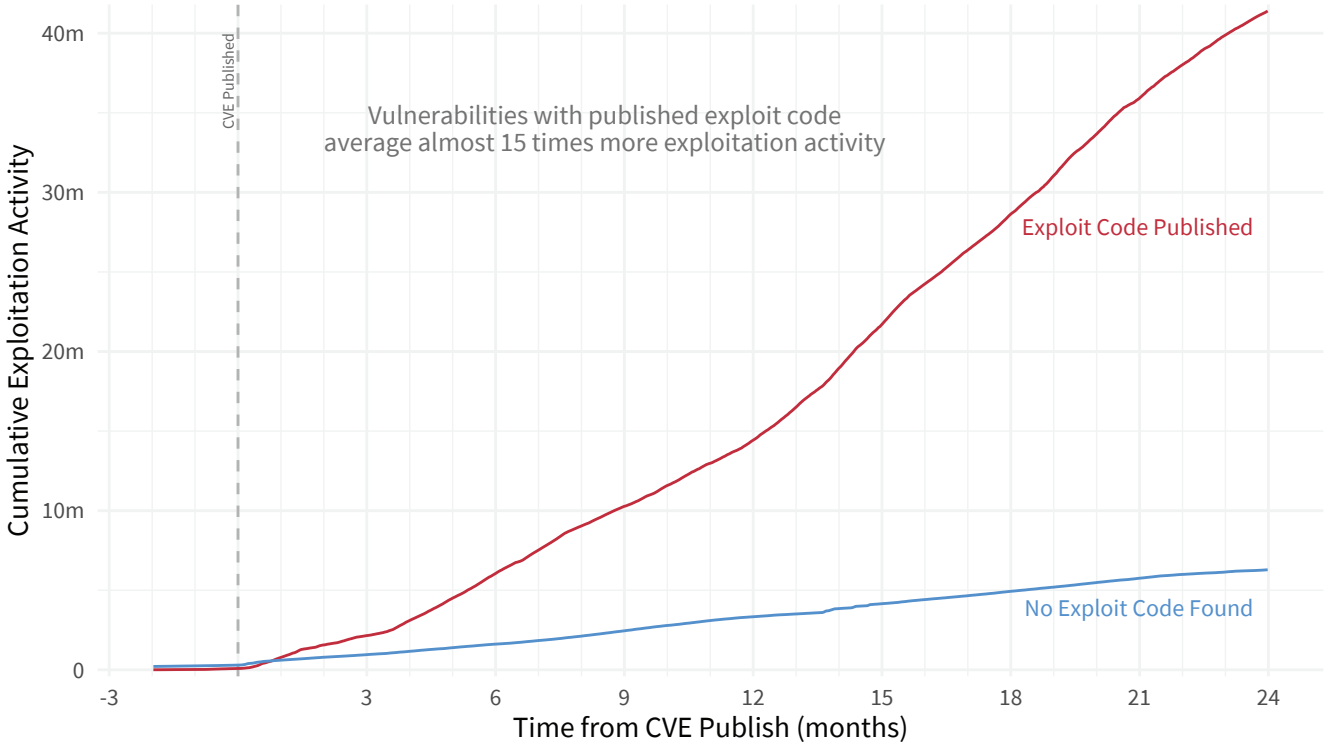

Figure 8: Overall exploitation activity detected for vulnerabilities with vs. without published exploit code

Based on the evidence presented in this section, we cannot reject this hypothesis. While "proving" a hypothesis isn't something a self-respecting scientist does, we do view this analysis as very compelling support for the conclusion that releasing exploits before patches—or dare we say publicly releasing them at all—ultimately helps attackers more than it helps defenders.

That important finding brings us full circle back to the notion of attacker-defender advantage. If premature exploitation pushes the advantage in favor of attackers, what other factors might impact this delicate balance? Can you guess where the next section is headed?

Attacker-Defender Advantage

Let's begin with a macro-level view of attacker-defender advantage. Figure 9 constructs a variant of Figure 1 without the pre- vs. post-exploit distinction. It also incorporates the full set of data we've collected on vulnerabilities exploited in the wild, which is why it shows a two-year time frame instead of one. And one more change—we decided to use CVE publication as the start date (dashed line) rather than patch release.†

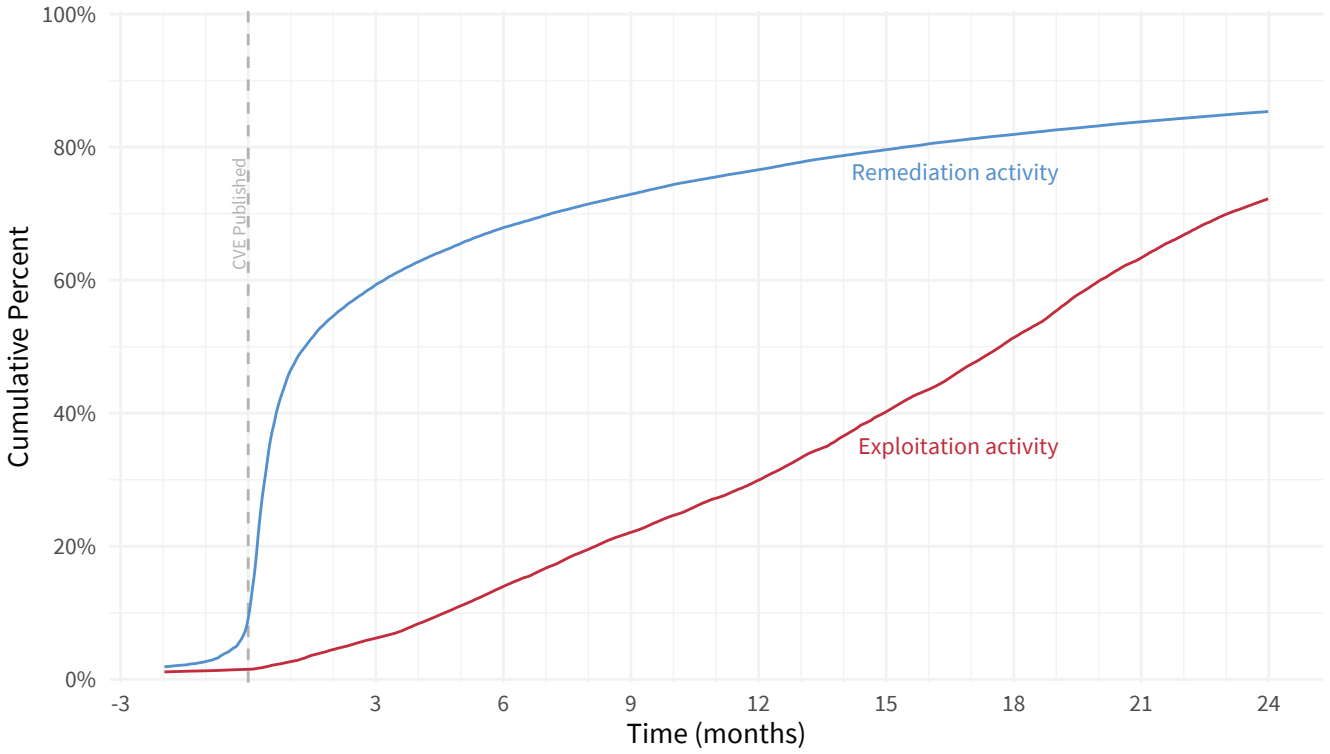

Figure 9: Overall attacker-defender advantage based on vulnerability remediation and exploitation timelines

Figure 9 offers a rare breath of fresh air for defenders. The blue remediation line starts and stays above the red exploitation line. That means defenders own the advantage the entire time by remediating vulnerabilities at a faster rate than attackers exploit them. If that rosy picture doesn't jive with your experience, you're not alone. It'll become apparent as we move forward that this overall view of attacker-defender advantage is more theoretical than practical.

The reality is there's a wide range of organizational, operational, and technical factors that alter attacker-defender advantage. We'd absolutely love to study that full spectrum, but our current dataset focuses entirely on that last aspect—technology. More specifically, the next section examines how various types of software affect the balance between attackers and defenders.

† *Two reasons: 1) We're not contrasting exploits before/after patch release in this section. 2) Not all CVEs have a known patch date, so eliminating that allows us to examine more vulnerabilities.*

Revisiting Remediation Velocity

After seven volumes of research, introducing topics that build upon older work becomes challenging because we can't assume everyone's read everything that's come before. Understanding factors influencing the defender curve in Figure 9 is a good example. In P2P Volume 3, we compared remediation velocity across a variety of dimensions, including major software vendors, industry, and organization size. We can't repeat it all here, but we selected and updated one chart from that report to create some continuity. If you want more backstory, Volume 3 eagerly awaits your in-depth review (as do all the others).

Figure 10 compares remediation velocity across major Common Platform Enumeration (CPE) vendors at three key milestones (25%, 50%, and 75% of vulns closed). The differences are dramatic. For instance, it takes over 40 times longer for organizations to address half their vulnerabilities affecting Linux and SAP software (~900 days) than to reach that same milestone with Google and Microsoft products (~22 days)! What's more, moving from 50% to 75% remediation takes multiple years for several vendors. While it is true that remediating vulnerabilities is an internal process, this serves as a good reminder that vendors, technology, and other external factors have a huge impact on remediation timelines.

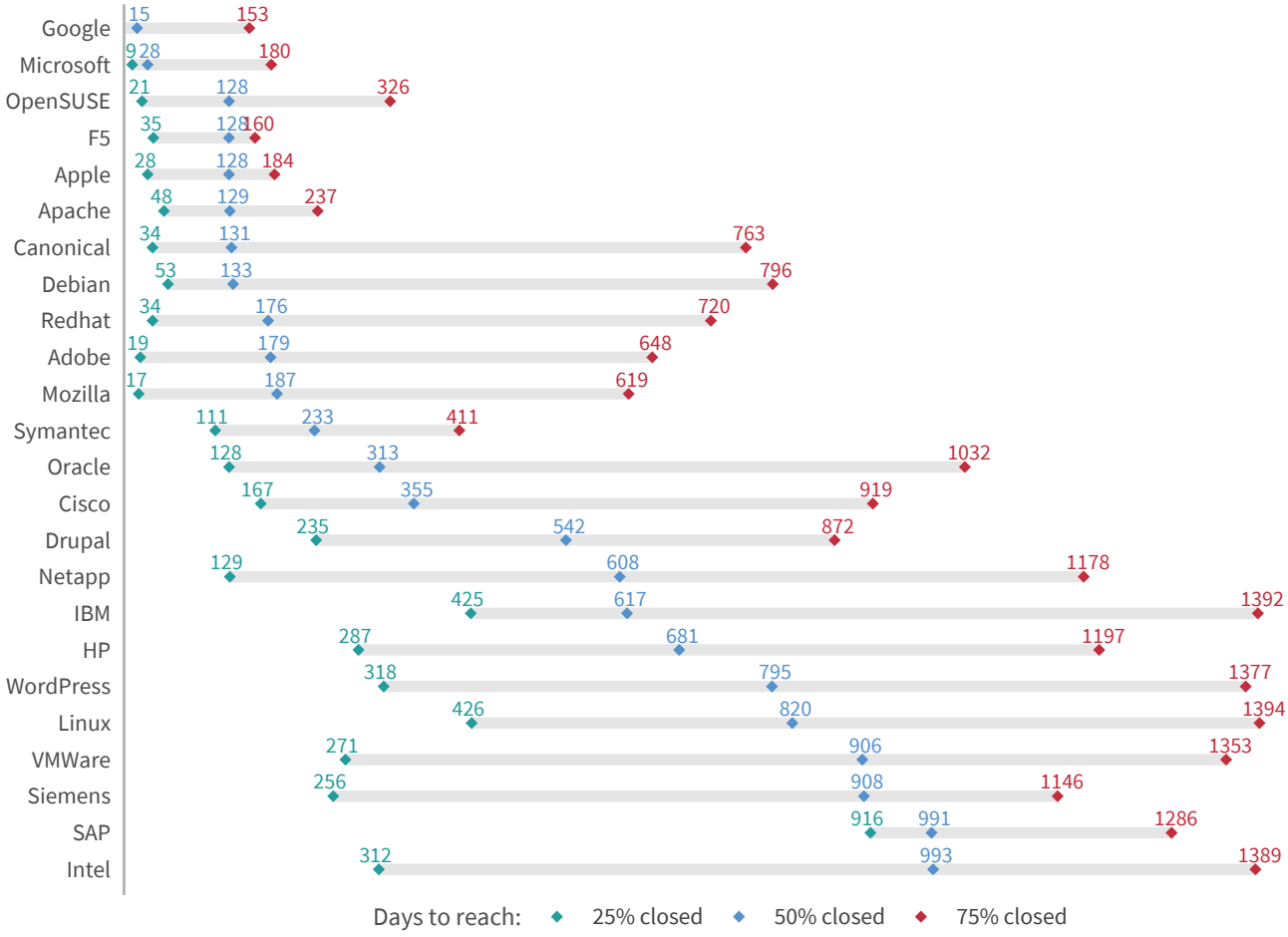

Figure 10: Comparing remediation velocity for major product vendors

While it is true that remediating vulnerabilities is an internal process, this serves as a good reminder that vendors, technology, and other external factors have a huge impact on remediation timelines.

Comparing Exploitation Measures

We're not quite ready to officially introduce exploitation metrics equivalent to those developed for measuring remediation performance in prior P2Ps. We'll get there eventually, though. For now, we'd like to casually explore a few things we can measure in the data that appear to influence the exploitation side of the attacker-defender dynamic.

Figure 11 plots a selection of 50† software products according to the proportion of CVEs with known exploit code (x-axis) and observed exploitation in the wild (y-axis). The products are colored according to the category of asset with which they are typically associated (unless the vendor is generally platform-independent).

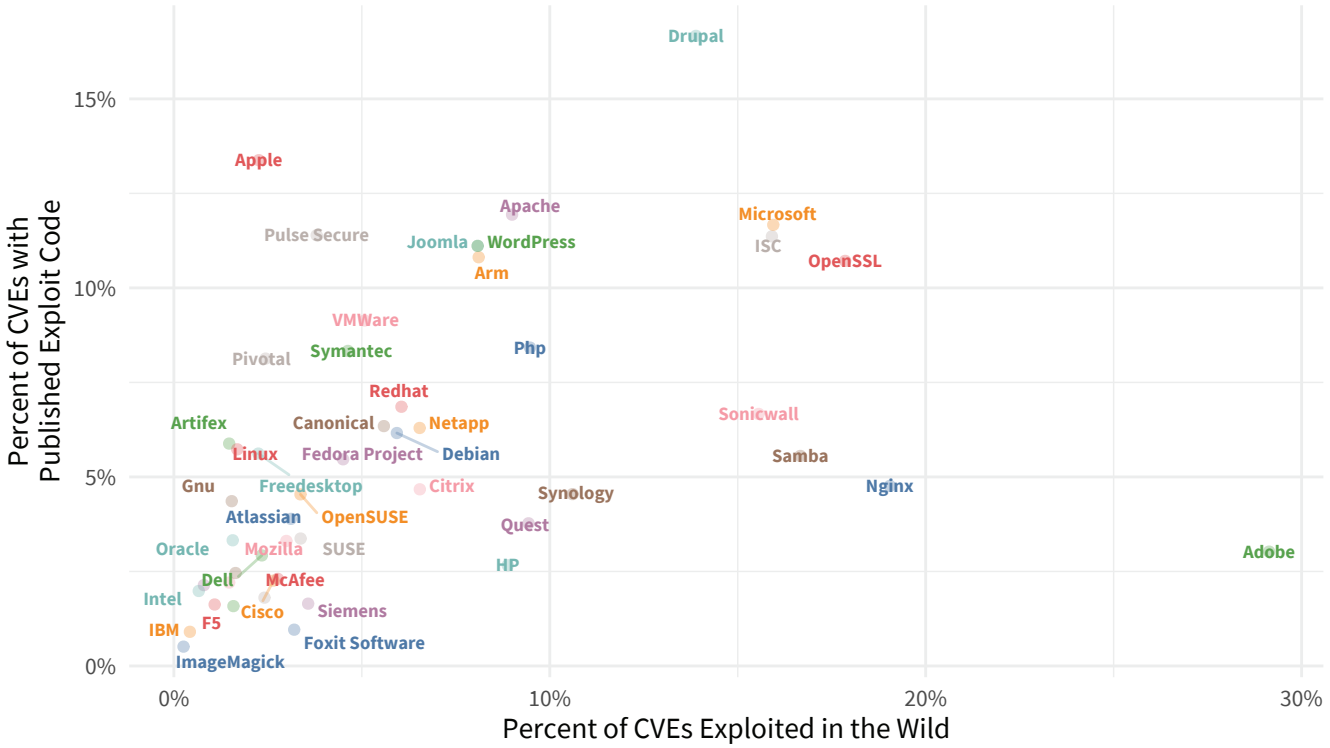

Figure 11: Major software vendors plotted according to percent of CVEs with exploit code and exploitation in the wild

There's obviously a relationship between exploits and exploitation, but it's fairly loose, and some products buck the trend altogether. Overall, we think Figure 11 says a lot about the asymmetrical nature of exploitation activity.

From the plot, it's clear that Adobe products are much more likely to be exploited in the wild than any other vendor. Nginx falls a distant second, amid a short list of vendors exhibiting 10% to 20% of CVEs exploited in the wild. Drupal claims the highest proportion of published CVEs with associated exploit code, followed by Apple. There's obviously a relationship between exploits and exploitation, but it's fairly loose, and some products buck the trend altogether (Adobe, Apple). Overall, we think Figure 11 says a lot about the asymmetrical nature of exploitation activity.

† *We factored in four variables to select CPE for inclusion: count of CVEs exploited, total exploitation in the wild, count of CVEs observed by vuln scanners, and total number of vulnerable assets.*

Figure 12 presents another interesting way to compare software exploitability. The horizontal axis places vendors based on the total number of vulnerabilities across all assets managed by the organizations in our dataset. So, those toward the right have a large (and vulnerable) digital footprint. The vertical axis measures the total amount of exploitation activity detected for each vendor. Products sitting high on this scale often fall within attackers' crosshairs.

We tossed a lot more vendors into Figure 12, making it one of those visuals best suited for exploring on your own. That probably involves locating software used in your environment. Naturally, that software mix will be different for everyone. For our part, we'll just mention the main quadrants as a useful way of dividing up the vendor landscape:

- Lower-Left: Less adopted; Less attacked
- Lower-Right: More adopted; Less attacked
- Upper-Right: More adopted; More attacked
- Upper-Left: Less adopted; More attacked

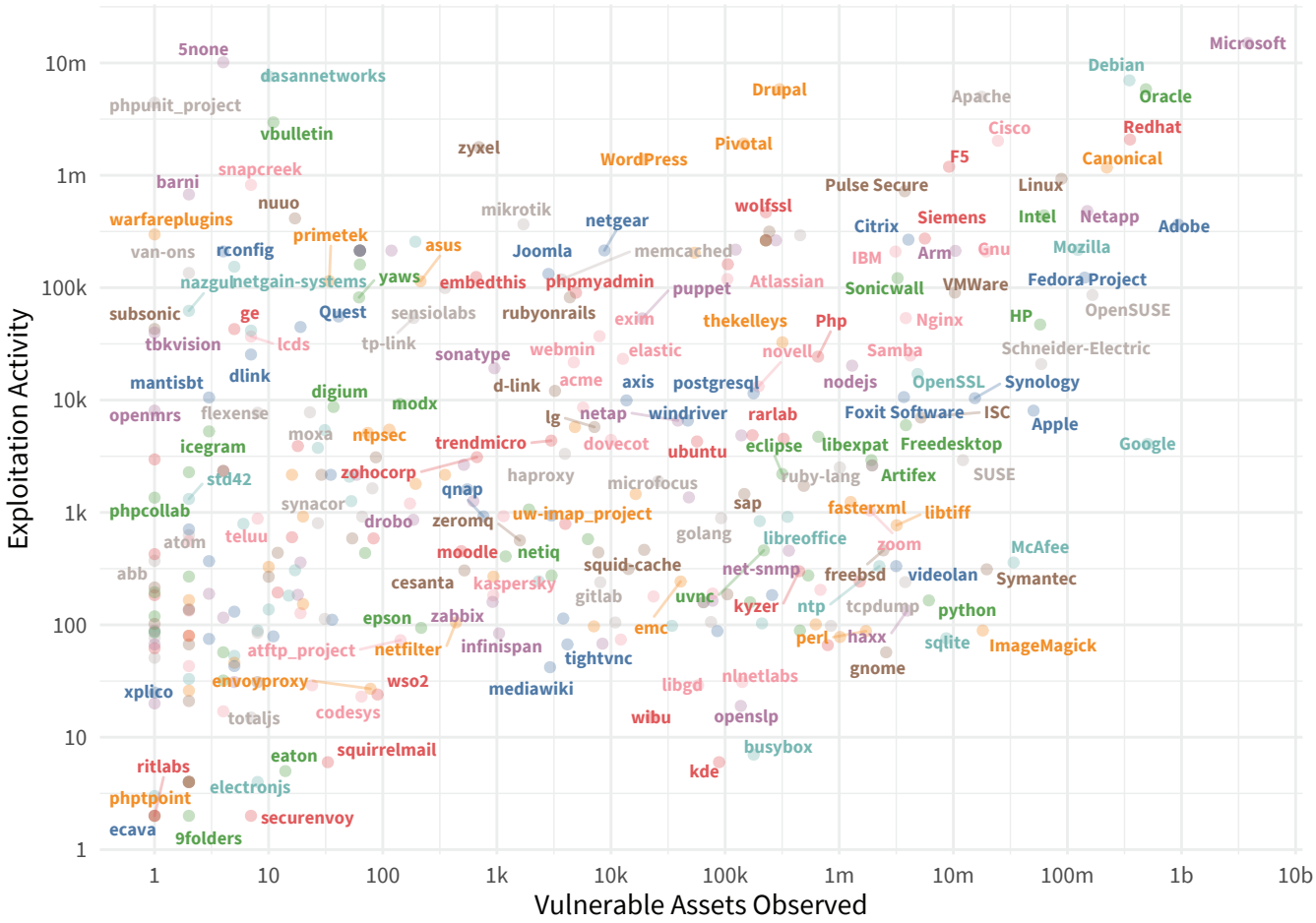

Figure 12: Major software vendors plotted according to the total number of vulnerabilities and exploitations observed

Here's an interesting question to ask yourself after perusing Figure 12: how is the software in your environment distributed across these quadrants? That ratio might reveal something useful about the exploitability of your infrastructure.

How is the software in your environment distributed across these quadrants? That ratio reveals something about the exploitability of your infrastructure.

Attacker-Defender Advantage by Software

The previous charts should help set the stage for the main event we present in this section. In essence, that involves creating remediation and exploitation curves and obtaining a measure of attacker-defender advantage for each CPE vendor. If that sounds as awesome to you as it does to us, behold as we unpack the incredible amount of information contained in Figure 13.

Let's do basic orientation first. The product vendors shown in Figure 13 were selected because they rank high in terms of total number of vulnerabilities detected and/or total amount of exploitation activity observed across organizations in our dataset. The bold red and blue lines represent the rate of exploitation and remediation, respectively, and the gray, dashed line marks the date of CVE publication. The faint lines in the background are there to aid quick comparisons to the overall average shown in Figure 9.

Figure 13 adds one super nifty feature beyond earlier charts, though—a single value capturing the relative degree of attacker-defender advantage exhibited by each vendor. Calculating that gets rather complicated, but the short story is that the value is the difference between the areas under each curve. So, if you see a blue positive number in the upper-left corner, it indicates some degree of defender advantage. A negative red number means attackers have the upper hand for that product set (rate of exploitation exceeds remediation).

Now that we know how to interpret Figure 13, we could literally fill pages with interesting commentary around it. But don't worry; we'll restrain ourselves and just hit the highlights.

One thing we noticed right away is that remediation rates vary much more than exploitation rates across vendors. To a certain extent, that's simply what happens when you average a bunch of things together. The result is just average. But we also think this indicates that, by and large, exploit activity is more random and constant than it is targeted and bursty. Attackers scouring the internet with automated tools to find vulnerable systems often aren't too choosy about which software they target. If there's a weakness, they'll exploit it.

Vendors do, on the other hand, heavily influence the rate of remediation. And on that note, let's give a big round of applause for Google and Microsoft, which earn the highest scores for defender advantage across their products. That's especially impressive because they have more products, more vulnerabilities, and a larger install base than most other vendors. Despite that large surface area of exposure, they're enabling defenders to remediate very quickly. Much could be said about this feat, but what come to mind first are their mature, coordinated disclosure programs and their practice of frequent, automatic updates.

Not all vendors are faring so well, however. Several, including Siemens and Drupal, exhibit a negative defender advantage. Both vendors can be found in the upper-right quadrant back in Figure 12, indicating they're relatively widely adopted and widely attacked. They're two very different technologies, however, which may be why the red and blue lines follow different paths. The remediation rate for Siemens' vulnerabilities falls near the bottom among vendors shown (but not as low as SAP) whereas Drupal exhibits one of the highest rates of exploitation. That's pretty much what we'd expect: industrial technology generally isn't as easy to update, and content management systems are notoriously easy to exploit.

Exploit activity is more random and constant than it is targeted and bursty.

Vendors do, on the other hand, heavily influence the rate of remediation.

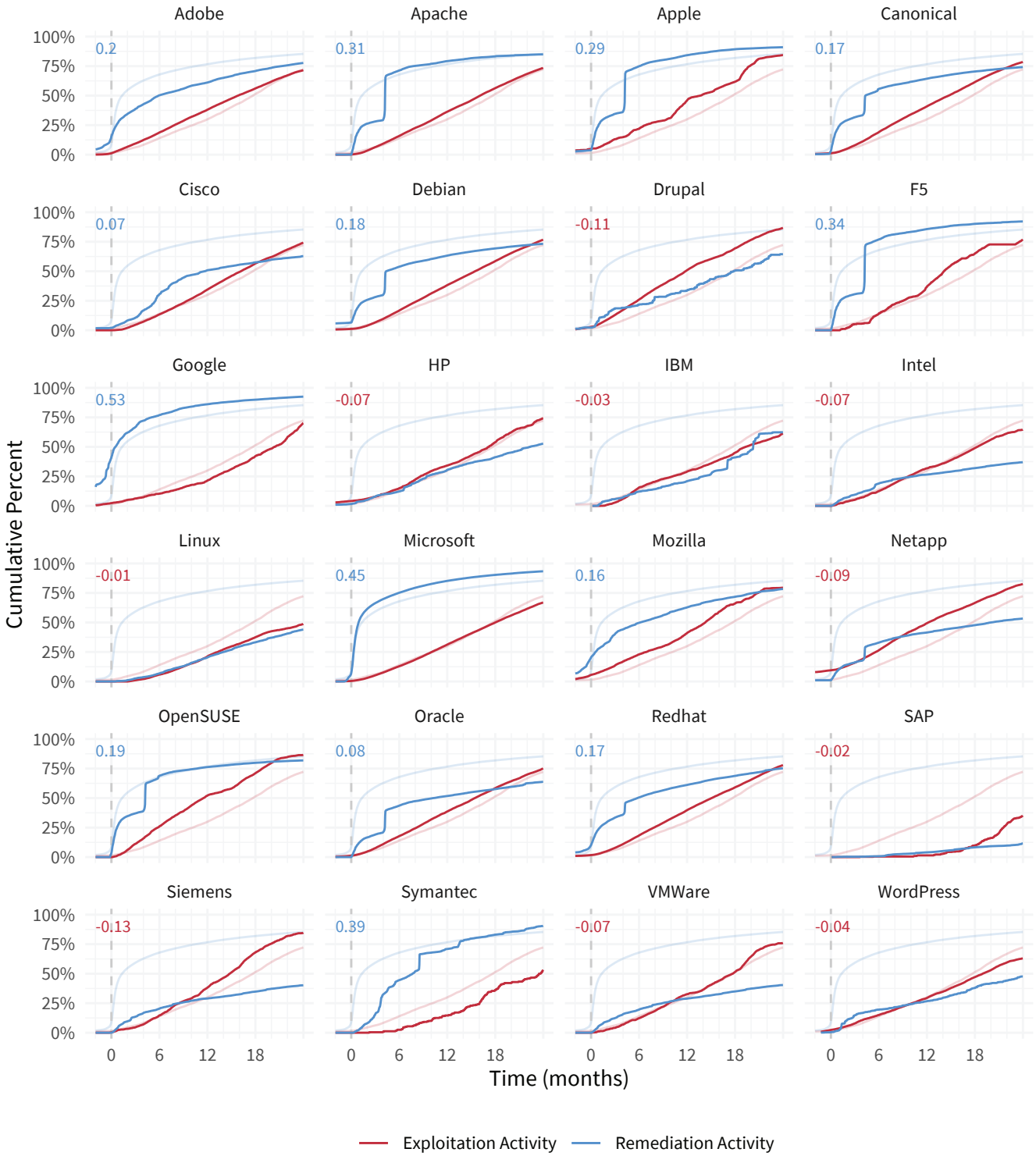

Figure 13: Comparison of attacker-defender advantage among major software vendors

All in all, Figure 13 demonstrates that different software products will have different strengths and weaknesses when it comes to maintaining defender advantage. That'll become even more obvious with our closing visualization.

Remediation of Google and Microsoft vulns outpaced exploitation the entire two-year period of study. Siemens and Wordpress maintained defender advantage for just 3 of 24 months.

What about CVSS?

Some of you reading this report might be wondering how the severity of vulnerabilities plays into attacker-defender advantage. The Common Vulnerability Scoring System (CVSS) is the best-known metric for this, so we'll take a quick gander at the data along this dimension in Figure 14.

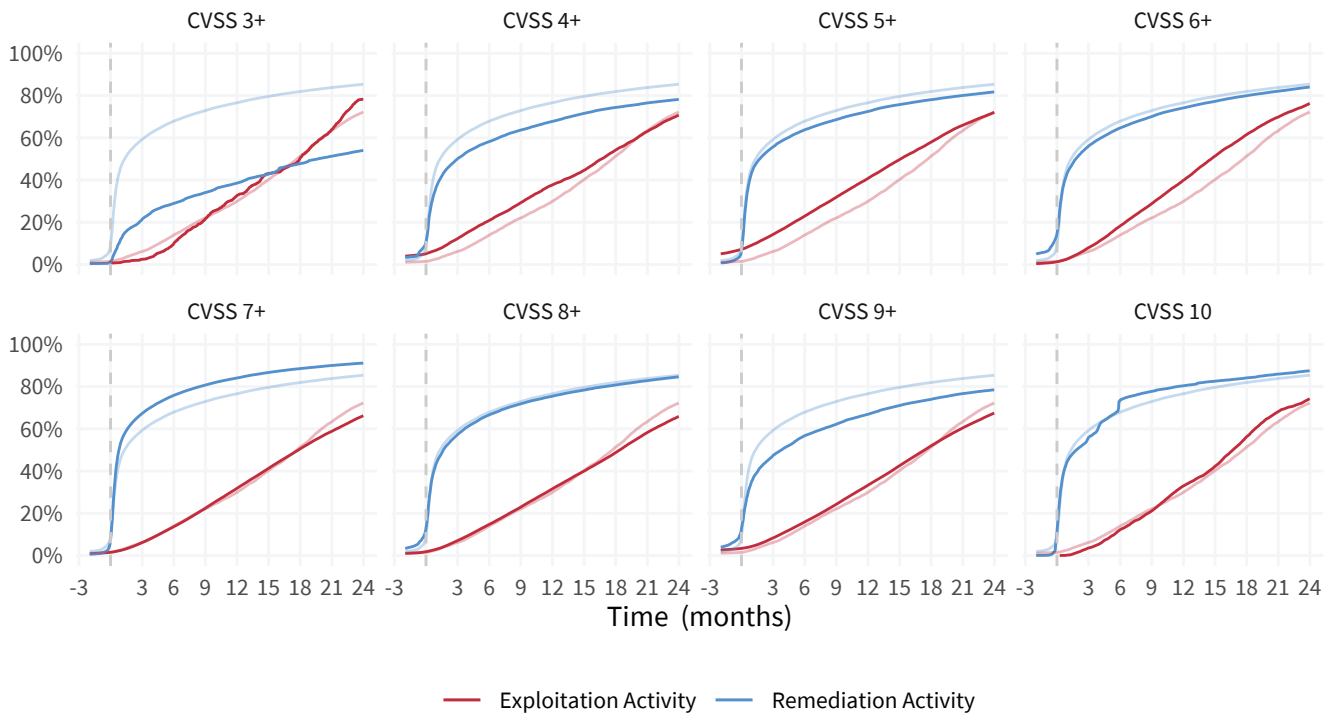

Figure 14: Comparison of attacker-defender advantage by CVSS score

We don't see any major revelations here, but there are a few observations of note. First, the red lines don't change that much, indicating that attackers generally don't give much credence to severity ratings when selecting vulnerabilities to target. This corroborates our earlier research into factors that predict exploitation in the wild.

It's also interesting the blue remediation lines don't change that much either, other than for the lowest of the low-severity vulns. Curiously, a CVSS score of 7 seems to elicit more urgency than 8, 9, and 10. That probably has a lot to do with the number of CVEs in each score range. This too lines up with results we've seen in prior volumes on this topic.

Figure 13 is special, but it's not quite worthy of a P2P grand finale visualization that ties off this volume and teases what's next in the series. However, we think Figure 15 is up to the task. It contains the same basic elements as Figure 13 with some extra flare:

1. It includes more vendors because dots are smaller than curves.
2. Speaking of dots, those are sized relative to the number of detected vulnerabilities.
3. Vendors are plotted according to remediation (x-axis) and exploitation (y-axis) rates.
4. The dashed lines separate over- and under-performers on each dimension.

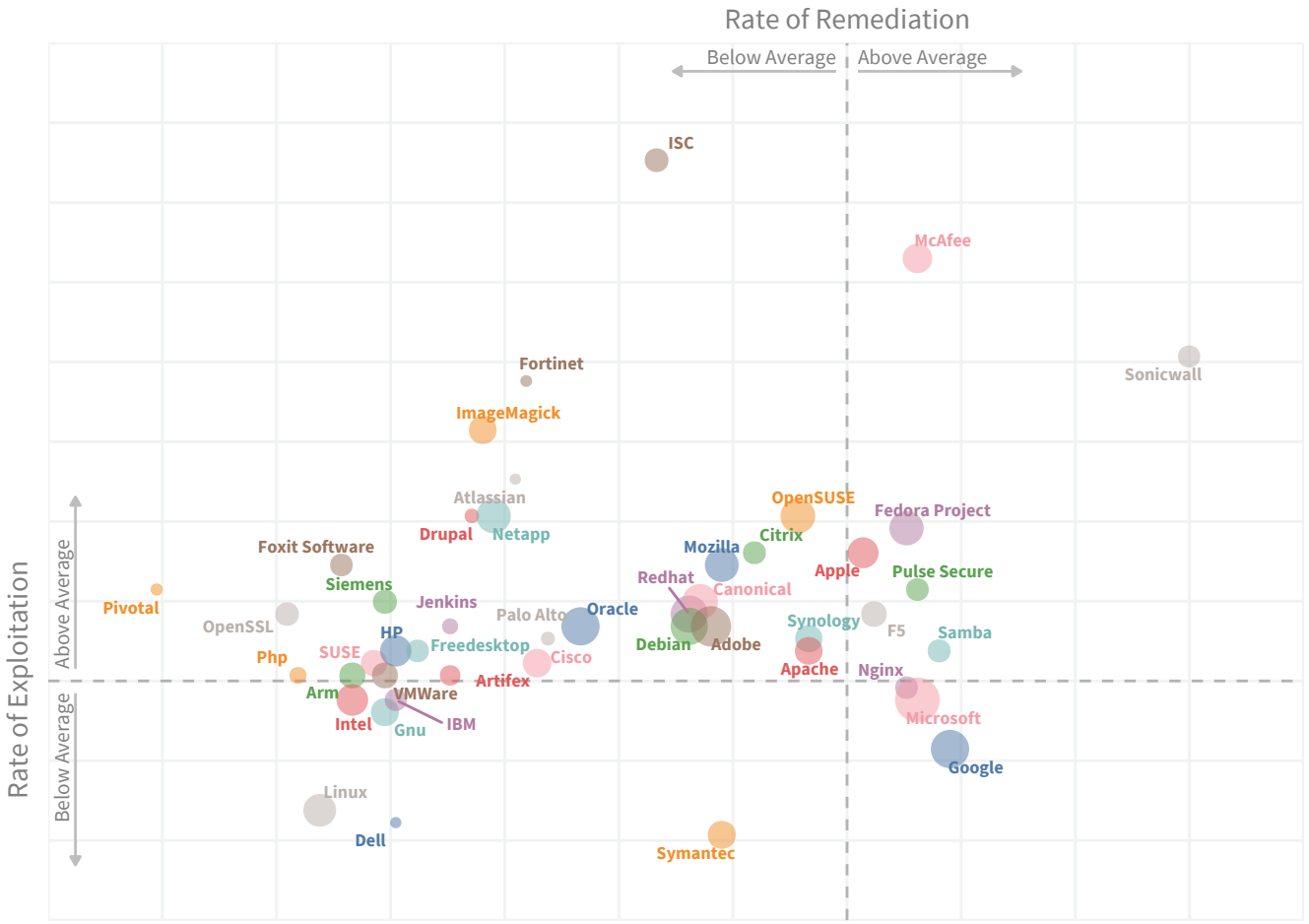

Figure 15: Relative measure of attacker-defender advantage among major software vendors

Based on these results, software products in the lower-right quadrant are inherently more conducive to maintaining defender advantage. Those in other quadrants—especially in the upper-right—will make that feat more challenging. That's not to say your security destiny is determined solely by the software in your environment. But Figure 15 definitely shows that VM programs which adapt to the strengths and weaknesses of their tech stack are best positioned to reduce risk efficiently.

How does an organization know if they're in such a position? Is it possible to determine the relative exploitability or remediability (yep; it's a word) of an entire organization? How does this factor into the probability of attacks and other risk measures? All good questions to which we hope to offer data-driven answers in a future installment of the P2P series. Until then, may your blue curve stay well above the red.

Evidence-Backed Recommendations

We'd like to conclude this report in similar fashion to the last volume, with some brief takeaways specific to vulnerability researchers, product vendors, and enterprise defenders.

For Researchers: All evidence points to coordinated disclosure putting defenders in the best position to gain advantage over attackers. In fact, our findings on the lag between patch availability and remediation suggest that it would benefit enterprise defenders even more if additional time was allowed between patch and exploit release in order to reduce exposure. To that end, we applaud Google Project Zero's recent policy change that gives an extra 30-day cushion for organizations to patch before full details of a vulnerability are publicly disclosed. The evidence supports this decision, and we hope others follow suit.

For Vendors: We encourage vendors to continue supporting coordinated disclosure of vulnerabilities as well as those who engage in it. That seems to work best when researchers are treated as assets rather than adversaries. Furthermore, we notice vendors with mature patch release processes and automatic updates (e.g., Google, Microsoft) tend to maximize advantage for defenders (see Figure 15). We've proposed a way to objectively measure the relative degree of attacker-defender advantage offered by product vendors, and we hope this will become a competitive differentiator that helps raise the bar for all.

For Defenders:

1. Vulnerabilities with exploit code are far more likely to see widespread exploitation in the wild than those without. That means the release of exploit code is a critical indicator of increased risk for your organization. To minimize exposure, continuously track exploit publication from multiple sources and prioritize remediation efforts based on that intelligence.

2. Understand the strengths and weaknesses of IT assets in your environment when it comes to exploitation and remediation. Some tend to be exploited more/less quickly and some appear inherently easier/harder to remediate. You'll find it much easier to maintain defender advantage when the assets in your environment help rather than hinder that goal.

3. Related to that last point, take advantage of features that aid defender advantage such as automatic updates. Sure, this always needs to be weighed against operational and other risks introduced by pushing patches without fully testing them first. But these decisions should also consider the benefits of reducing the timeframe of exposure offered by such features.