Project Zero

News and updates from the Project Zero team at Google

Wednesday, July 29, 2020

Detection Deficit: A Year in Review of 0-days Used In-The-Wild in 2019

Posted by Maddie Stone, Project Zero

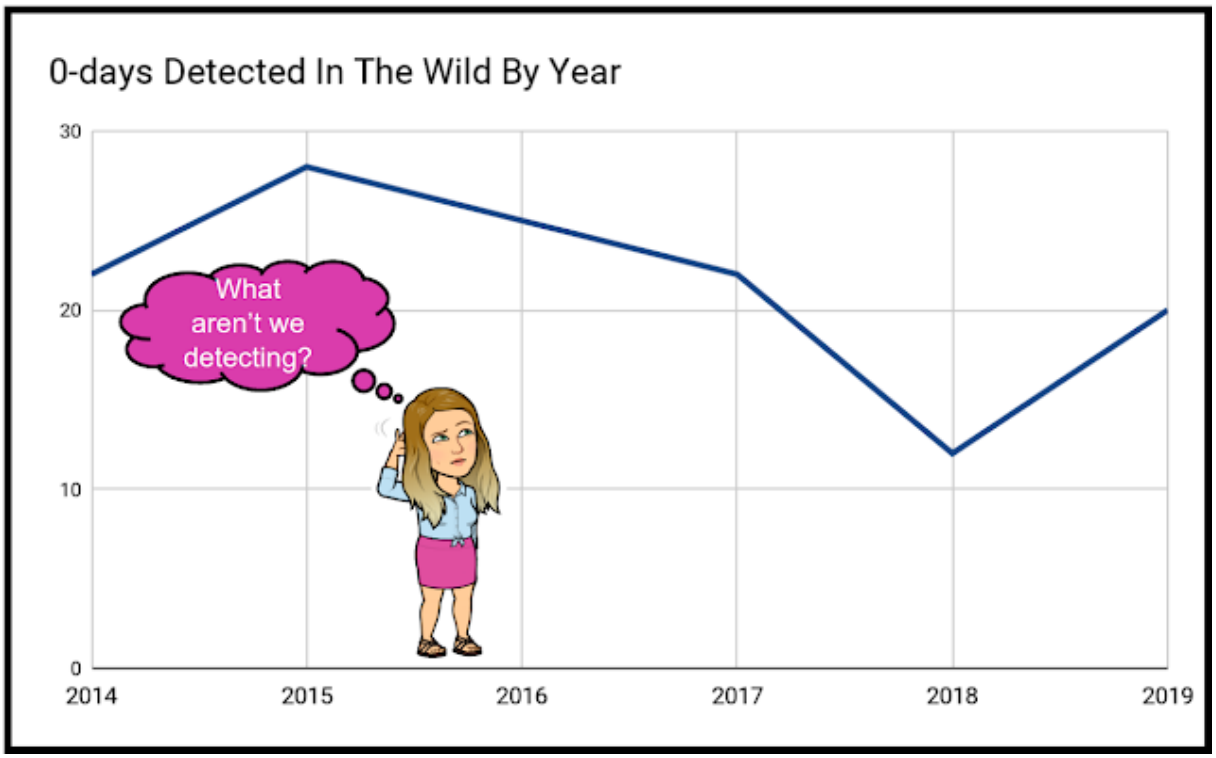
In May 2019, Project Zero released our tracking spreadsheet for 0-days used "in the wild" and we started a more focused effort on analyzing and learning from these exploits. This is another way Project Zero is trying to make zero-day hard. This blog post synthesizes many of our efforts and what we've seen over the last year. We provide a review of what we can learn from 0-day exploits detected as used in the wild in 2019. In conjunction with this blog post, we are also publishing another blog post today about our root cause analysis work that informed the conclusions in this Year in Review. We are also releasing 8 root cause analyses that we have done for in-the-wild 0-days from 2019.

When I had the idea for this "Year in Review" blog post, I immediately started brainstorming the different ways we could slice the data and the different conclusions it may show. I thought that maybe there'd be interesting conclusions around why use-after-free is one of the most exploited bug classes or how a given exploitation method was used in Y% of 0-days or… but despite my attempts to find these interesting technical conclusions, over and over I kept coming back to the problem of the detection of 0-days. Through the variety of areas I explored, the data and analysis continued to highlight a single conclusion: **As a community, our ability to detect 0-days being used in the wild is severely lacking to the point that we can't draw significant conclusions due to the lack of (and biases in) the data we have collected.**

The rest of the blog post will detail the analyses I did on 0-days exploited in 2019 that informed this conclusion. As a team, Project Zero will continue to research new detection methods for 0-days. We hope this post will convince you to work with us on this effort.

The Basics

In 2019, 20 0-days were detected and disclosed as exploited in the wild. This number, and our tracking, is scoped to targets and areas that Project Zero actively researches. You can read more about our scoping here. This seems approximately average for years 2014-2017 with an uncharacteristically low number of 0-days detected in 2018. *Please note that Project Zero only began tracking the data in July 2014 when the team was founded and so the numbers for 2014 have been doubled as an approximation.*



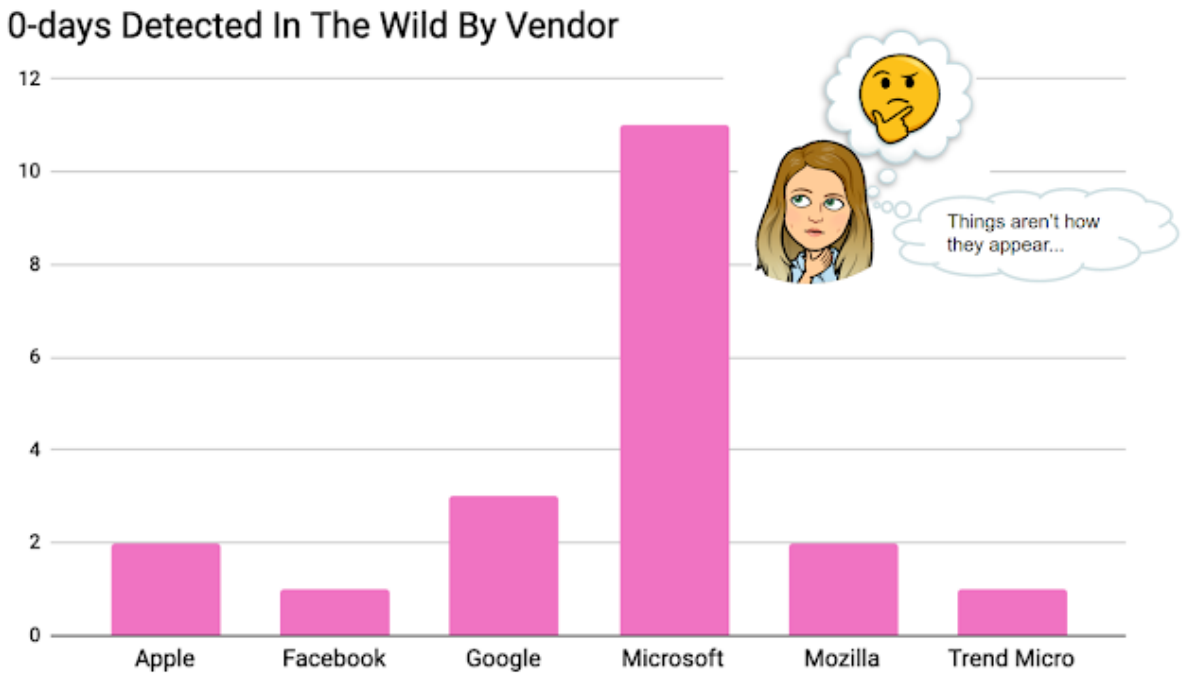
The largely steady number of detected 0-days might suggest that defender detection techniques are progressing at the same speed as attacker techniques. That could be true. Or it could not be. The data in our spreadsheet are only the 0-day exploits that were detected, not the 0-day exploits that were used. As long as we still don't know the true detection rate of all 0-day exploits, it's very difficult to make any conclusions about whether the number of 0-day exploits deployed in the wild are increasing or decreasing. For example, if all defenders stopped detection efforts, that could make it appear that there are no 0-days being exploited, but we'd clearly know that to be false.

All of the 0-day exploits detected in 2019 are detailed in the Project Zero tracking spreadsheet here.

0-days by Vendor

One of the common ways to analyze vulnerabilities and security issues is to look at who is affected. The breakdown of the 0-days exploited in 2019 by vendor is below. While the data shows us that almost all of the big platform vendors have at least a couple of 0-days detected against their products, there is a large disparity. Based on the data, it *appears* that Microsoft products are targeted about 5x more than Apple and Google products. Yet Apple and Google, with their iOS and Android products, make up a huge majority of devices in the world.

While Microsoft Windows has always been a prime target for actors exploiting 0-days, I think it's more likely that we see more Microsoft 0-days due to detection bias. Because Microsoft has been a target before some of the other platforms were even invented, there have been many more years of development into 0-day detection solutions for Microsoft products. Microsoft's ecosystem also allows for 3rd parties, in addition to Microsoft themself, to deploy detection solutions for 0-days. The more people looking for 0-days using varied detection methodologies suggests more 0-days will be found.



Microsoft Deep-Dive

For 2019, there were 11 0-day exploits detected in-the-wild in Microsoft products, more than 50% of all 0-days detected. Therefore, I think it's worthwhile to dive into the Microsoft bugs to see what we can learn since it's the only platform we have a decent sample size for.

Of the 11 Microsoft 0-days, only 4 were detected as exploiting the latest software release of Windows . All others targeted earlier releases of Windows, such as Windows 7, which was originally released in 2009. Of the 4 0-days that exploited the latest versions of Windows, 3 targeted Internet Explorer, which, while it's not the default browser for Windows 10, is still included in the operating system for backwards compatibility. This means that 10/11 of the Microsoft vulnerabilities targeted legacy software.

Out of the 11 Microsoft 0-days, 6 targeted the Win32k component of the Windows operating system. Win32k is the kernel component responsible for the windows subsystem, and historically it has been a prime target for exploitation. However, with Windows 10, Microsoft dedicated resources to locking down the attack surface of win32k. Based on the data of detected 0-days, none of the 6 detected win32k exploits were detected as exploiting the latest Windows 10 software release. And 2 of the 0-days (CVE-2019-0676 and CVE-2019-1132) only affected Windows 7.

Even just within the Microsoft 0-days, there is likely detection bias. Is legacy software really the predominant targets for 0-days in Microsoft Windows, or are we just better at detecting them since this software and these exploit techniques have been around the longest?

CVE	Windows 7 SP1	Windows 8.1	Windows 10	Win 10 1607	Win 10 1703	Win 10 1803	Win 10 1809	Win 10 1903	Exploitation of Latest SW Release?	Component
CVE-2019-0676	X	X	X	X	X	X	X		Yes (1809)	IE
CVE-2019-0808	X								N/A (1809)	win32k
CVE-2019-0797		X	X	X	X	X	X		Exploitation Unlikely (1809)	win32k
CVE-2019-0703	X	X	X	X	X	X	X		Yes (1809)	Windows SMB
CVE-2019-0803	X	X	X	X	X	X	X		Exp More Likely (1809)	win32k
CVE-2019-0859	X	X	X	X	X	X	X		Exp More Likely (1809)	win32k
CVE-2019-0880	X	X	X	X	X	X	X	X	Exp More Likely (1903)	splwow64
CVE-2019-1132	X								N/A (1903)	win32k
CVE-2019-1367	X	X	X	X	X	X	X	X	Yes (1903)	IE
CVE-2019-1429	X		X	X	X	X	X	X	Yes (1903)	IE
CVE-2019-1458	X	X	X	X					N/A (1909)	win32k

Internet Explorer JScript 0-days CVE-2019-1367 and CVE-2019-1429

While this blog post's goal is not to detail each 0-day used in 2019, it'd be remiss not to discuss the Internet Explorer JScript 0-days. CVE-2019-1367 and CVE-2019-1429 (and CVE-2018-8653 from Dec 2018 and CVE-2020-0674 from Feb 2020) are all variants of each other with all 4 being exploited in the wild by the same actor according to Google's Threat Analysis Group (TAG).

Our root cause analysis provides more details on these bugs, but we'll summarize the points here. The bug class is a JScript variable not being tracked by the garbage collector. Multiple instances of this bug class were discovered in Jan 2018 by Ivan Fratric of Project Zero. In

December 2018, Google's TAG discovered this bug class being used in the wild (CVE-2018-8653). Then in September 2019, another exploit using this bug class was found. This issue was “fixed” as CVE-2019-1367, but it turns out the patch didn’t actually fix the issue and the attackers were able to continue exploiting the original bug. At the same time, a variant was also found of the original bug by Ivan Fratric (P0 1947). Both the variant and the original bug were fixed as CVE-2019-1429. Then in January 2020, TAG found another exploit sample, because Microsoft’s patch was again incomplete. This issue was patched as CVE-2020-0674.

A more thorough discussion on variant analysis and complete patches is due, but at this time we’ll simply note: The attackers who used the 0-day exploit had 4 separate chances to continue attacking users after the bug class and then particular bugs were known. If we as an industry want to make 0-day harder, we can’t give attackers four chances at the same bug.

Memory Corruption

63% of 2019’s exploited 0-day vulnerabilities fall under memory corruption, with half of those memory corruption bugs being use-after-free vulnerabilities. Memory corruption and use-after-free’s being a common target is nothing new. “Smashing the Stack for Fun and Profit”, the seminal work describing stack-based memory corruption, was published back in 1996. But it’s interesting to note that almost two-thirds of all detected 0-days are still exploiting memory corruption bugs when there’s been so much interesting security research into other classes of vulnerabilities, such as logic bugs and compiler bugs. Again, two-thirds of detected 0-days are memory corruption bugs. While I don’t know for certain that that proportion is false, we can't know either way because it's easier to detect memory corruption than other types of vulnerabilities. Due to the prevalence of memory corruption bugs and that they tend to be less reliable then logic bugs, this could be another detection bias. Types of memory corruption bugs tend to be very similar within platforms and don’t really change over time: a use-after-free from a decade ago largely looks like a use-after-free bug today and so I think we may just be better at detecting these exploits. Logic and design bugs on the other hand rarely look the same because in their nature they’re taking advantage of a specific flaw in the design of that specific component, thus making it more difficult to detect than standard memory corruption vulns.

Even if our data is biased to over-represent memory corruption vulnerabilities, memory corruption vulnerabilities are still being regularly exploited against users and thus we need to continue focusing on systemic and structural fixes such as memory tagging and memory safe languages.

More Thoughts on Detection

As we’ve discussed up to this point, the same questions posed in the team's original blog post still hold true: “What is the detection rate of 0-day exploits?” and “How many 0-day exploits are used without being detected?”.

We, as the security industry, are only able to review and analyze 0-days that were detected, not all 0-days that were used. While some might see this data and say that Microsoft Windows is exploited with 0-days 11x more often than Android, those claims cannot be made in good faith. Instead, I think the security community simply detects 0-days in Microsoft Windows at a much higher rate than any other platform. If we look back historically, the first anti-viruses and detections were built for Microsoft Windows rather than any other platform. As time has continued, the detection methods for Windows have continued to evolve. Microsoft builds tools and techniques for detecting 0-days as well as third party security companies. We don’t see the same plethora of detection tools on other platforms, especially the mobile platforms, which means there’s less likelihood of detecting 0-days on those

platforms too. An area for big growth is detecting 0-days on platforms other than Microsoft Windows and what level of access a vendor provides for detection..

Who is doing the detecting?

Another interesting side of detection is that a single security researcher, Clément Lecigne of the Google's TAG is credited with 7 of the 21 detected 0-days in 2019 across 4 platforms: Apple iOS (CVE-2019-7286, CVE-2019-7287), Google Chrome (CVE-2019-5786), Microsoft Internet Explorer (CVE-2019-0676, CVE-2019-1367, CVE-2019-1429), and Microsoft Windows (CVE-2019-0808). Put another way, we could have detected a third less of the 0-days actually used in the wild if it wasn't for Clément and team. When we add in the entity with the second most, Kaspersky Lab, with 4 of the 0-days (CVE-2019-0797, CVE-2019-0859, CVE-2019-13720, CVE-2019-1458), that means that two entities are responsible for more than 50% of the 0-days detected in 2019. If two entities out of the entirety of the global security community are responsible for detecting more than half of the 0-days in a year, that's a worrying sign for how we're using our resources. . The security community has a lot of growth to do in this area to have any confidence that we are detecting the majority of 0-days exploits that are used in the wild.

Out of the 20 0-days, only one (CVE-2019-0703) included discovery credit to the vendor that was targeted, and even that one was also credited to an external researcher. To me, this is surprising because I'd expect that the vendor of a platform would be best positioned to detect 0-days with their access to the most telemetry data, logs, ability to build detections into the platform, "tips" about exploits, etc. This begs the question: are the vendor security teams that have the most access not putting resources towards detecting 0-days, or are they finding them and just not disclosing them when they are found internally? Either way, this is less than ideal. When you consider the locked down mobile platforms, this is especially worrisome since it's so difficult for external researchers to get into those platforms and detect exploitation.

"Clandestine" 0-day reporting

Anecdotally, we know that sometimes vulnerabilities are reported surreptitiously, meaning that they are reported as just another bug, rather than a vulnerability that is being actively exploited. This hurts security because users and their enterprises may take different actions, based on their own unique threat models, if they knew a vulnerability was actively exploited. Vendors and third party security professionals could also create better detections, invest in related research, prioritize variant analysis, or take other actions that could directly make it more costly for the attacker to exploit additional vulnerabilities and users if they knew that attackers were already exploiting the bug. If all would transparently disclose when a vulnerability is exploited, our detection numbers would likely go up as well, and we would have better information about the current preferences and behaviors of attackers.

0-day Detection on Mobile Platforms

As mentioned above, an especially interesting and needed area for development is mobile platforms, iOS and Android. In 2019, there were only 3 detected 0-days for all of mobile: 2 for iOS (CVE-2019-7286 and CVE-2019-7287) and 1 for Android (CVE-2019-2215). However, there are billions of mobile phone users and Android and iOS exploits sell for double or more compared to an equivalent desktop exploit according to Zerodium. We know that these exploits are being developed and used, we're just not finding them. The mobile platforms, iOS and Android, are likely two of the toughest platforms for third party security solutions to deploy upon due to the "walled garden" of iOS and the application sandboxes of both platforms. The same features that are critical for user security also make it difficult for third parties to deploy on-device detection solutions. Since it's so difficult for non-vendors to

deploy solutions, we as users and the security community, rely on the vendors to be active and transparent in hunting 0-days targeting these platforms. Therefore a crucial question becomes, how do we as fellow security professionals incentivize the vendors to prioritize this?

Another interesting artifact that appeared when doing the analysis is that CVE-2019-2215 is the first detected 0-day since we started tracking 0-days targeting Android. Up until that point, the closest was CVE-2016-5195, which targeted Linux. Yet, the only Android 0-day found in 2019 (AND since 2014) is CVE-2019-2215, which was detected through documents rather than by finding a zero-day exploit sample. Therefore, no 0-day exploit samples were detected (or, at least, publicly disclosed) in all of 2019, 2018, 2017, 2016, 2015, and half of 2014. Based on knowledge of the offensive security industry, we know that that doesn't mean none were used. Instead it means we aren't detecting well enough and 0-days are being exploited without public knowledge. Therefore, those 0-days go unpatched and users and the security community are unable to take additional defensive actions. Researching new methodologies for detecting 0-days targeting mobile platforms, iOS and Android, is a focus for Project Zero in 2020.

Detection on Other Platforms

It's interesting to note that other popular platforms had no 0-days detected over the same period: like Linux, Safari, or macOS. While no 0-days have been publicly detected in these operating systems, we can have confidence that they are still targets of interest, based on the amount of users they have, job requisitions for offensive positions seeking these skills, and even conversations with offensive security researchers. If Trend Micro's OfficeScan is worth targeting, then so are the other much more prevalent products. If that's the case, then again it leads us back to detection. We should also keep in mind though that some platforms may not need 0-days for successful exploitation. For example, this blogpost details how iOS exploit chains used publicly known n-days to exploit WebKit. But without more complete data, we can't make confident determinations of how much 0-day exploitation is occurring per platform.

Conclusion

Here's our first Year in Review of 0-days exploited in the wild. As this program evolves, so will what we publish based on feedback from you and as our own knowledge and experience continues to grow. We started this effort with the assumption of finding a multitude of different conclusions, primarily "technical", but once the analysis began, it became clear that everything came back to a single conclusion: we have a big gap in detecting 0-day exploits. Project Zero is committed to continuing to research new detection methodologies for 0-day exploits and sharing that knowledge with the world.

Along with publishing this Year in Review today, we're also publishing the root cause analyses that we completed, which were used to draw our conclusions. Please check out the blog post if you're interested in more details about the different 0-days exploited in the wild in 2019.