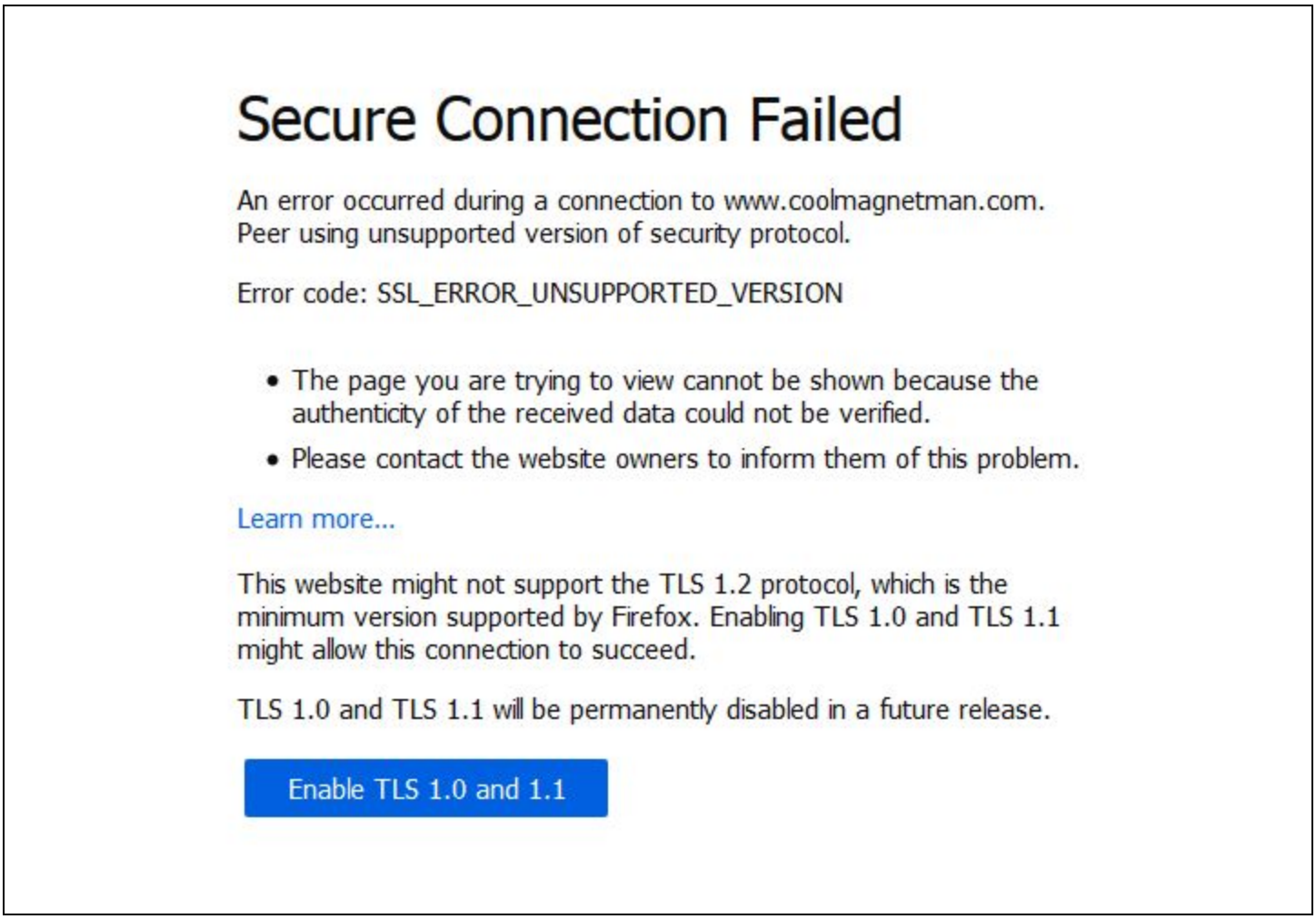
Security Now! #784 - 09-15-20

BlindSide & BLURtooth

This week on Security Now!

This week we look at the Chrome browser's proactive technology which is designed to punish abusive ads. We also look at the last hurrah for exploiting IE and Adobe Flash users, some Microsoft Edge updates, last Tuesday's Microsoft Patch-a-Palooza, Zoom's new implementation of two factor authentication, that very bad WordPress File Manager attack two-weeks out, the new Raccoon attack against TLS and a quick SpinRite update. Then, we conclude with a look at two newly discovered attacks named BlindSide and BLURtooth.



Browser News

Chrome getting tough on abusive ads

In a posting on Github, Google engineer John Delaney has spelled out the Chromium project's intentions regarding abusive ads:
https://github.com/johnivdel/heavy-ad-intervention/blob/master/README.md#proposed-thresholds

But first of all, modern web pages are a jungle. So how does Chromium determine what's an advertisement and what isn't? This comes down to something known as Ad Tagging.

Chromium is able to detect some ads and the resources they load in the browser. This enables the browser to measure the size, performance, and count of ads displayed to our users. It also allows the browser to intervene on the user's behalf when ads run counter to the user's interest (e.g., by using excessive resources or by engaging in abusive behavior.

The ad detection infrastructure is called Ad Tagging. Ad Tagging works by matching resource requests against a filter list to determine if they're ad requests. Any requests matching the filter are tagged as ads. Further, requests (and some DOM elements such as iframes) made on behalf of previously tagged scripts are also tagged as ads by the AdTracker. An iframe will be marked as an ad iframe if its url matches the filter list, if tagged script is involved in the creation of the iframe, or if its parent frame is an ad iframe. The main frame of a page will never be tagged as an ad. Any request made within an ad iframe is considered an ad resource request.

Drilling down on this one level, we learn that the Subresource Filter loads the filter list and matches url requests against it. The list is distributed via the component updater. This same list and component is used for blocking ads on abusive sites and those that violate the Better Ads Standard.

Each subresource request in the render process is processed by the subresource filter before the request is sent. If it matches the list, the ResourceRequest is tagged as an ad.

Each subframe navigation request is processed in the browser process by the SubresourceFilter and if the final URL in the navigation matches the list then the render process is told (via the SubresourceFilterAgent) and the iframe is marked as an ad iframe.

The AdTracker keeps track of each script subresource that is considered an ad. Any time that a new resource is requested or an iframe is created, V8's stack is scanned and if any of the known ad scripts are in the stack then the resource request or iframe is considered an ad.

Okay... So now that ads are identified as such, how are they treated differently?

This brings us to John Delaney's "Heavy Ad Intervention Explainer". Establishing a bit of background, John explains that: "A small fraction of ads on the web use an egregious amount of system resources. These poorly performant ads (whether intentional or not) harm the user's browsing experience by making pages slow, draining device battery, and consuming mobile data (for those without unlimited plans).

In these egregious cases, the browser can unload the offending ads to protect the individual's device resources. This is a strong intervention that is meant to safeguard the user's resources with low risk because unloading an ad is unlikely to result in loss of functionality of the page's main content."

Examples of observed ad behaviors that are intended to be discouraged:

- Ads that mine cryptocurrency
- Ads that load large, poorly compressed images
- Ads that load large video files before a user gesture
- Ads that perform expensive operations in javascript, such as decoding video files, or CPU timing attacks

Google notes that it is not their intention to discourage any specific ad creative formats, such as display video ads.

So... The user agent will unload ads that use an egregious amount of network bandwidth or CPU usage. We define egregious as using more of a resource than 99.9% of ads as measured by the browser. Only ads that have not been interacted with by the user will be unloaded.

All unloaded frames will be notified via an ***Intervention Report*** that the intervention occurred. (We'll talk about the Intervention Reporting mechanism in a moment.) This feedback is necessary to help advertisers or their ad technology vendors to identify and fix ads that are triggering this intervention.

The classification of ads is left to the discretion of the user agent. For example, Chrome detects ads using its AdTagging feature — which we just discussed. An advertisement is considered "**heavy**" if it has not been clicked on by the user and meets any of the following criteria:

- Used the main thread for more than 60 seconds total
- Used the main thread for more than 15 seconds in any 30 second window (50% utilization over 30 seconds)
- Used more than 4 megabytes of network bandwidth to load resources

The thresholds above were inspired by the IAB's Lean Standard but chosen by looking at Chrome's metrics at the 99.9th percentile of network and CPU usage in ads. Threshold numbers were rounded to be readable and memorable. Numbers were picked inclusive of all ads seen by Chrome's AdTagging, which is better at capturing display ads than native ads.

A total CPU usage threshold is used in addition to a peak usage threshold to discourage ads from using just below the peak window usage, which still drains the user's battery over time.

Intervening at the above thresholds is projected to save 12.8% of the network usage by ad creatives, and 16.1% of all CPU usage by ad creatives, with most of that value going to individual users. Thresholds are agnostic to platform -- desktop or mobile makes no difference -- so that creative authors can easily know if their ad would be subject to intervention or not. Thresholds may need to change as the web ecosystem and common device profiles evolve.

There's one concept that we've never discussed on the podcast, which is an "Intervention Report"...

Modern web browsers support a "Reporting API" which allows them to send back asynchronous event reports to a resource's source server. As we know, a web page, or code on a web page, will make a query to a remote web server for content. This might be an advertisement. One of the headers in the remote web server's reply can be "Report-To:". Its corresponding string value is a json formatted string to which reports of any subsequent issues with this resource can be sent at any later time, subject to a recipient's requested expiration. So, if a JavaScript crash were to occur when it was executed, and if the JavaScript reply contained a "Report-To:" header, the web browser could inform the sending server of this. Or, in our present case, if an advertisement breaks one of these behavioral rules and is therefore unloaded, it's now possible using this Reporting API to inform the remote website that this has occurred.

Report-To: { "url": "https://example.com/reports", "max_age": 10886400 }

There's a fun way to play with this stuff if you have Chrome 84 or later. There are several relevant Chrome flags:

- Enable chrome://flags/#enable-heavy-ad-intervention
- Disable chrome://flags/#heavy-ad-privacy-mitigations

Setting chrome://flags/#enable-heavy-ad-intervention to Enabled activates the new behavior, but by default there is some noise and variability added to the thresholds to protect user privacy. Setting chrome://flags/#heavy-ad-privacy-mitigations to Disabled prevents this, meaning the restrictions are applied deterministically, purely according to the limits. This should make debugging and testing easier.

The last hurrah for IE & Flash exploits

Malwarebyte Labs has issued a warning about their observation of a sudden large surge in attempted malware attacks leveraging oldie but goodie vulnerabilities in Internet Explorer and Abode Flash. These attacks are being observed on sites featuring, shall we say, "highly explicit adult sexual content."

One of the vulnerabilities from last year "CVE-2019-0752":
This vulnerability allows remote attackers to execute arbitrary code on vulnerable installations of Microsoft Internet Explorer. User interaction is required to exploit this vulnerability in that the target must visit a malicious page or open a malicious file. The specific flaw exists within the handling of script commands that set certain properties of DOM objects. By performing actions in script, an attacker can trigger a type confusion condition. An attacker can leverage this vulnerability to execute code in the context of the current process.

... And for anyone unlucky enough to visit one of the infected websites with an unpatched IE, one of two well known exploit kits will be downloaded and executed.

And if that one doesn't get ya, the site will also try to leverage an even older flaw, from 2018, that existed in Adobe's still-not-quite-dead-yet (it's just a flesh wound) Flash Player.

https://blog.malwarebytes.com/social-engineering/2020/09/malvertising-campaigns-come-back-in-full-swing/

The trouble is that a surprising number of people — hopefully none listening to this podcast — are still venturing out into the world, pushing a copy of Internet Explorer ahead of them... and it's just no longer up to the job.

The group behind the attacks, named "Malsmoke", has operated on a scale far above other similar cybercrime operations and has, according to Malwarebytes, which has been tracking Malsmoke's attacks, abused "practically all adult ad networks." Most of the time, the group has managed to place malicious ads (so-called malverts) only on mid-tier adult portals. But they recently "hit the jackpot" when they managed to sneak malverts on xHamster, which (and this was news to me) is one of the biggest adult video portals in existence, and one of the biggest sites on the Internet [you've got to be kidding], with **billions** of visitors each month.

Chromium Edge on Win10: Forcing the issue

There was also some news that flashed by, noting that Microsoft was going to begin pushing their Chromium-based Edge onto people's Win10 machines whether they asked for it or not. Everyone should have asked for it long ago -- there's just no downside unless some misguided enterprise wrote custom code that only runs on the original Edge. In that case, it's time to fix that. In any event, I have no problem with having everyone moved over to a Chromium based engine.

Edge: Enabling "Ask me..." for each download

And speaking of Microsoft Edge and downloads, as we were also last week, the original Classic Edge offered a few additional post file download options which have been missing from the new Chromium Edge, specifically, "Delete" and "Copy download link", both which might come in handy and are not yet offered by other browsers. Since Edge users had been missing them, they will soon be arriving in the new Edge. At this time, the Edge Dev and Canary versions greater than 87.0.629.0 have this, and once they make it into the general release, we all be able to enable them.

They need to be enabled at the moment, and it's unclear whether that will always be the case, but the switch to enable them is definitely good to have anyway, since it's the "Ask me what to do with each download." mode which, I think, everyone should use. If you want the default it only requires an additional click on "OK", but the fact that it presents a big prompting SaveAs dialog that clearly informs you of the event, makes it very worthwhile.

Just go to: edge://settings/downloads and turn on "Ask me what to do with each download."

Security News

Patch Tuesday Palooza!

Last Tuesday was September's 129 vulnerability patch drop and as those in the industry have admonished, try to be kind to your Windows admins who will be scrambling to juggle the impact of last Tuesday's 23 CRITICAL vulnerabilities while working to resolve the bugs caused by patching them.

This was not the all-time greatest patch month, but it did tie with this past June's 129 record breaker. Microsoft provided patches to repair 23 critical flaws in Windows and related products, 105 Important flaws, and a lone moderate vulnerability.

While there were fortunately no 0-days known to be under active exploitation, there were quite a few interesting vulnerabilities that could be exploited remotely.

For the enterprise, CVE-2020-16875 would likely grab the attention of an administrator, being a memory corruption vulnerability in Exchange that can allow a remote attacker to perform remote code execution as the SYSTEM user merely by sending an specially crafted email to an Exchange server. Whoopsie.

Dustin Childs, a researcher at Trend Micro's Zero-Day Initiative (ZDI), wrote in his analysis last Tuesday: "That is about the worst-case scenario for Exchange servers. We have seen the previously patched Exchange bug CVE-2020-0688 used in the wild, and that requires authentication. We'll likely see this one in the wild soon. This should be your top priority." And as we know, the danger, now that this has been patched, is that clever bad guys will examine the fix, then design and launch attacks before the Exchange admins can get their systems updated. So don't dally on this one.

Another critical RCE vulnerability that should be prioritized for patching is CVE-2020-1210. That one exists in SharePoint, due to a failure to check an application package's source markup. It rates 9.9 out of 10 on the CVSS severity scale. Satnam Narang, a staff research engineer at Tenable wrote: "To exploit this flaw, an attacker would need to be able to upload a SharePoint application package to a vulnerable SharePoint site. This vulnerability is reminiscent of a similar SharePoint remote code-execution flaw, CVE-2019-0604, that has been exploited in the wild by threat actors since at least April 2019." This month there are a total of seven RCE bugs being fixed in SharePoint. And only one of them, CVE-2020-1460, requires any authentication. So, yes... Again we see that patched vulnerabilities are irresistible to attackers who know that they will almost certainly find some laggards at high-value targets.

Justin Knapp, the product marketing manager at Automox, pointed to another critical RCE vulnerability (this one with an 8.4 rating) which was just fixed in the Windows Graphic Device Interface (CVE-2020-1285). This one arises because of the way GDI handles objects in memory, providing both web-based and file-sharing attack scenarios that could introduce multiple vectors for an attacker to gain control of a system. In the web-based attack scenario, an attacker would merely need to craft a website designed to exploit the vulnerability, then convince users to view the website." Not a high bar on that one.

And then there's CVE-2020-1129, another remote code execution flaw in Microsoft Windows Codecs Library (an 8.8 rating). As we know, state-of-the-art codecs are quiTe difficult to make perfect. Yet perfect they must be. Any program that can cause Microsoft HEVC codec to be invoked can be the exploit entry. An attacker could execute code on a victim machine by convincing someone to view a weaponized video clip. The flaw exists within the parsing of HEVC streams such that a crafted HEVC video file can trigger an overflow of a fixed-length stack-based buffer.

And then there's CVE-2020-0922 which describes itself as "Microsoft COM for Windows Remote Code Execution Vulnerability can be exploited by luring a user to a site with malicious JavaScript." Hmmmmm... How would we know that a site has malicious JavaScript? Best not to find out.

And the mysteriously described CVE-2020-0908 is: "Windows Text Service Module Remote Code Execution Vulnerability can be exploited by tricking a user to visiting a site that contains malicious user-provided content or advertisements." Uhhhhhhhh... Pretty much all sites contain user-provided content or advertisements.

So, by now, everyone is safely patched and updated against all of these nightmares... right? Let's hope that COVID isn't still causing outages in enterprise vulnerability management. This month's disclosures could pack a wallop.

Excessive SSD Defragging also fixed

One last little reminder: Recall that something broke in Windows that was causing it to forget that it had ever defragged its mass storage drives. So it was doing so, needlessly and inducing unnecessary wear, every time the system was booted. It was also issuing "Trim" commands to spinning hard drives that do not support that command because they're not SSDs.

Well, all of this was also fixed last Tuesday. So anyone who might have manually disabled the automatic maintenance of your SSDs to suppress this unnecessary wear, can now, after installing last Tuesday's updated, safely re-enable automatic drive maintenance.

Surprisingly, the bug of the mis-issued "Trim" commands to spinning hard drives persists. But it's not critical. It simply fails and posts a note into the system's error logs.

The WordPress File Manager flaw... two weeks downstream

Our first coverage of this was last week, where I introduced the problem by saying: "It's not good when a 0-day flaw is discovered being actively exploited in an extremely popular plug-in for Wordpress. And it's also somewhat jarring that we keep covering exactly such news."

And... here we are, again.

Last week we covered the startup of attacks on vulnerable WorePress sites, which began within an hour of the public disclosure of the vulnerability. As usual, the race was on to get into sites via the vulnerability before the sites could be updated to close the hole.

Researchers with the WordPress security firm Defiant spotted more than 1.7 million sites being probed by bad guys between September 1st and the 3rd. In Defiant's updated report that was published a week later, last Thursday the 10th, threat analyst, Ram Gall, wrote that the attackers have not stopped their siege -- anything but, in fact. Now the number of WordPress sites being targeted has jumped to 2.6 million.

Multiple groups of bad guys are currently targeting the File Manager vulnerability, though Defiant has noted that there are a particular two from among the many who have seen the most success in deploying their malware on vulnerable sites.

One of the two is "Bajatax", a Morocco-based attacker known for stealing user credentials from specific e-commerce websites. In this instance, once this attacker compromises a WordPress site, he injects malicious code that harvests and exfiltrates user credentials via Telegram on any login, later to be sold to the highest bidder.

The second attacker has been observed to inject a pair of backdoors, one into a randomized folder and another onto the site's webroot. Both are camouflaged as .ico files to reduce the likelihood that the site's admin will find both and curtail the attacker's access to the site. As we have long said, once a server, a site, or a machine has been compromised it can never be fully trusted again.

The PHP infector that's being used by the second attacker is a variant of an infection that's been previously used to deploy cryptominers and run SEO spam campaigns via compromised sites.

And Defiant has observed both of these attackers working to block other attackers' exploit attempts by password protecting the exploitable "connector.minimal.php" file on the sites that they've infected.

Defiant's people said that their site cleaning team had cleaned a number of sites compromised by this File Manager vulnerability, and in many cases discovered malware placed there by multiple attackers. Obviously, by attackers who did not think to close the door behind them. But, thanks to those first two being proactive about blocking others, and collectively employing several thousand source IP addresses in their attacks, those two have been the most successful.

Overall, and somewhat incredibly, Defiant's researchers have monitored attacks attempting to exploit this vulnerability originating from more than 370,000 separate IP addresses.

"Zoom" ... now with 2FA

Last Thursday the 10th, Zoom announced the availability of TOTP-standard two-factor authentication, or SMS, or a phone call for logging into Zoom.

https://blog.zoom.us/secure-your-zoom-account-with-two-factor-authentication/

Administrators are able to choose how pervasive they wish 2FA to be for users of the account. Zoom's instructions say:

1. Sign in to the Zoom Dashboard
2. In the navigation menu, click Advanced, then Security.
3. Make sure the "Sign in with Two-Factor Authentication" option is enabled.
4. Select one of three options to enable 2FA for:
    a. Enable (require) 2FA for all users in the account.
    b. Enable (require) 2FA for roles with the specified roles.
       Click "Select specified roles", choose the roles, then click OK.
    c. Enable (require) 2FA for users that are in the specified groups.
       Click the pencil icon, choose the groups, then click OK.
5. Click 'Save' to confirm your 2FA settings.

Raccoon

So, one headline reads "New Raccoon Attack Could Let Attackers Break SSL/TLS Encryption" but a more sober headline reads "Raccoon attack allows hackers to break TLS encryption 'under certain conditions'" and then further notes: "The Raccoon attack is described as "really hard to exploit" and its conditions as "rare.""

When I was originally gathering topics to research for today's podcast, my first working title was "BlindSide, BLURtooth & Raccoon." But after digging into "Raccoon" I decided that it wouldn't be fair to lump it in with BlindSide and BLURtooth because those two are both actual problems, which is a bar that Raccoon doesn't begin to clear. But it is something that happened, so it bears mentioning, if only to put it into perspective.

https://raccoon-attack.com/
https://raccoon-attack.com/RacoonAttack.pdf

Their paper's is titled: "Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DHE"

They summarize this work by writing:

> Diffie-Hellman key exchange (DHKE) is a widely adopted method for exchanging cryptographic key material in real-world protocols like TLS-DHE. Past attacks on TLS-DHE focused on weak parameter choices or missing parameter validation. The confidentiality of the computed DH share, the premaster secret, was never questioned; DHKE is used as a generic method to avoid the security pitfalls of TLS-RSA. We show that due to a subtle issue in the key derivation of all TLS-DHE cipher suites in versions up to [and including] TLS 1.2, the premaster secret of a TLS-DHE session may, under certain circumstances, be leaked to an adversary. Our main result is a novel side-channel attack, named Raccoon attack, which exploits a timing vulnerability in TLS-DHE, leaking the most significant bits of the shared Diffie-Hellman secret.
>
> The root cause for this side channel is that the TLS standard encourages non-constant-time processing of the DH secret. If the server reuses ephemeral keys, this side channel may allow an attacker to recover the premaster secret by solving an instance of the Hidden Number Problem. The Raccoon attack takes advantage of uncommon DH modulus sizes, which depend on the properties of the used hash functions. We describe a fully feasible remote attack against

an otherwise-secure TLS configuration: OpenSSL with a 1032-bit DH modulus. Fortunately, such moduli are not commonly used on the Internet.

Furthermore, with our large-scale scans we have identified implementation-level issues in production-grade TLS implementations that allow for executing the same attack by directly observing the contents of server responses, without resorting to timing measurements.

Research is always good. But even the researchers were quoted saying: "The vulnerability is really hard to exploit and relies on very precise timing measurements and on a specific server configuration to be exploitable. The attacker needs to be close to the target server to perform high-precision timing measurements. The victim connection needs to use ephemeral DH and the server also needs to reuse ephemeral keys. And finally, the attacker needs to observe the original connection."

In other words, useful research, but nothing for us to worry about in the real world. All that said, Microsoft, Mozilla, OpenSSL and F5 Networks have all released security updates to block Raccoon attacks. So, as I said... useful academic research.

And by the way, don't Google "Raccoon Attack" for additional information — those little guys are much scarier than the digital attack that bears their name.

SpinRite

I'm down to a very few, but not quite zero, remaining edge cases with the mass storage benchmark on a few specific pieces of hardware. But, I also wanted to begin getting GRC's new web forums ready-to-go so that they would be ready when the code was. So, I spent some time over the weekend working to finish bringing them up. The last thing that needed finalizing is to get SQRL logon working on that domain. When I wrote GRC's server-side support for SQRL, I made it flexible enough to handle any domain, but I didn't build in support for multiple simultaneous domains. "sqrl.grc.com" & "forums.grc.com" are separate authentication domains as far as SQRL is concerned. Anyone using SQRL is asked to confirm the domain they're logging in to. So that should appear as "sqrl.grc.com" for the SQRL forums and "forums.grc.com" for the GRC forums. So, I added support to GRC's server-side implementation to handle any number of explicit subdomains of a parent domain. Rasmus Vind, the terrific web developer who knows the XenForo system and PHP upside down and backwards and I spent a bit of time yesterday, figuring out what might need to be updated at his end. I had updated to the latest release of the XenForo system, and many plug-ins needed to be tweaked after that, so it might be related. But I expect to have that behind me within another day or so. Then it'll be back to getting the benchmark finalized and ready for far more broader testing! :)

BlindSide & BLURtooth

BlindSide
A team consisting of five super-sharp researchers at ETH Zurich, the Stevens Institute of Technology and the University of Amsterdam, including our old friend professor Herbert Bos, has just managed to oh-so-cleverly leverage Spectre-style processor performance optimizations in such a way that, first, they bypass any attempts to mitigate the processor's Spectre-style vulnerabilities, and then leverage those vulnerabilities to successfully perform so-called BROP — "Blind Return Oriented Programming" — in the face of the best KASLR "Kernel Address Space Layout Randomization" while completely suppressing any "wrong layout guess" crashes.

Okay, so let's back up a bit.

A previous work by researchers at Stanford University was titled "Hacking Blind." The Abstract of their paper describes what they achieved, as follows:

https://www.scs.stanford.edu/brop/bittau-brop.pdf

> We show that it is possible to write remote stack buffer overflow exploits without possessing a copy of the target binary or source code, against services that restart after a crash. This makes it possible to hack proprietary closed-binary services, or open-source servers manually compiled and installed from source where the binary remains unknown to the attacker. Traditional techniques are usually paired against a particular binary and distribution where the hacker knows the location of useful gadgets for Return Oriented Programming (ROP). Our Blind ROP (BROP) attack instead remotely finds enough ROP gadgets to perform a write system call and transfers the vulnerable binary over the network, after which an exploit can be completed using known techniques. This is accomplished by leaking a single bit of information based on whether a process crashed or not when given a particular input string. BROP requires a stack vulnerability and a service that restarts after a crash. The attack works against modern 64-bit Linux with address space layout randomization (ASLR), no-execute page protection (NX) and stack canaries.

So, as we've often noted, when a system contains a vulnerability, attempts to exploit that vulnerability more often than not simply crash. In fact, that's what fuzzers do. They discover potentially weaponizable flaws by crashing a system that should not be crashable. Then, humans examine the execution path that the "fuzzing" triggered to see whether it might be exploitable.

And, as we know, one of the keys to spotting when a system might be under active attack is when the system's logs suddenly show that normally-reliable processes are suddenly crashing and auto-restarting. That might just be buggy code. But why would it choose to suddenly start happening? It could be the result of someone attempting, and repeatedly failing, to "guess" at the location of a known-vulnerable module in an OS kernel that employs KASLR whose kernel modules are randomized to occupy a different location for each boot... specifically to thwart this sort of "hit or miss" guessing attack.

Okay. So, with that background, what this team has managed to do is to figure out a way of completely neutering the powerful and arguably vital protections provided by KASLR by leveraging a system's Intel processor Spectre optimizations to repeatedly probe for the location of a known kernel bug in memory without ever crashing the system.

Their paper is titled: "Speculative Probing: Hacking Blind in the Spectre Era"
https://download.vusec.net/papers/blindside_ccs20.pdf

They've posted a video of their Proof-of-Concept code running on Linux. It's this week's GRC.SC shortcut, so, "http://grc.sc/784". That shortcut will bounce you over to a 4-minute YouTube video where you will watch their code running and obtaining ROOT. It's a bit chilling.

https://grc.sc/784
https://youtu.be/m-FUIZiRN5o

We've often quoted Bruce Schneier saying "Attacks never get weaker, they only get stronger." So here, after about 21 months of awareness and exploration of Spectre and what it means, we see it finally being very cleverly leveraged to perpetrate an extremely practical attack.

BLURtooth
So, the headlines for BLURtooth read:

- "BLURtooth vulnerability lets attackers defeat Bluetooth encryption"
- "Bluetooth Bug Opens Devices to Man-in-the-Middle Attacks"
- "BLURtooth vulnerability lets attackers overwrite Bluetooth authentication keys"
- "New Unpatched Bluetooth Flaw Lets Hackers Easily Target Nearby Devices"

BLURtooth is the result of insufficiently strict requirements appearing in previous Bluetooth specifications (from 4.0 through 5.0). Some tightening recommendations appeared in the minor 5.1 update, released in January of 2019. Technically, the mistake was in allowing for cross-transport keying of pairing associations. Devices that support both Low Energy (BLE) and Basic Rate/Enhanced Data Rate (BR/EDR) transport methods are known as "dual-mode" devices. And the earlier specs, as I said, allowed for those two differing transports to affect each other's pairing keys. And this was designed deliberately because it has its own names and abbreviation: Cross-Transport Key Derivation (CTKD). It turns out, as originally implemented — and as presently implemented by most dual-mode devices — it's not secure.

CTKD pairing allows the devices to pair once using either transport method while generating both the BR/EDR and LE Long Term Keys (LTK) without needing to pair a second time. Once again we get bitten by the desire for convenience because dual-mode devices using CTKD are able to overwrite the original Long Term Key or Link Key (LK) in cases where that transport was enforcing a higher level of security.

So, for example, the Bluetooth specification introduced so-called "Just Works" pairing for those instances where easy communications but less security is all that's needed. So, for example, fitness trackers, BLE-enabled jewelry, devices used to track pet behavior, and other technologies that do not deal with sensitive information such as credit card or health-related data.

But "Just Works" is part of BLE even when devices also support the deliberately far more secure BR/EDR pairing. As we know, one of the security enforcements for pairing is requiring a time-limited, human-triggered or some out-of-band communication to enable high-security pairing. But, two separate security research groups found that it's possible to leverage the simple, unsupervised, non-secure pairing offered by “Just Works” and use the CTKD (Cross-Transport Key Derivation) to break into and insecurely re-key the secure transport. Whoopsie.

There is no doubt that there are places where this would be a problem. The implementers of Bluetooth-connected systems might have deliberately implemented both transports under the entirely reasonable presumption that the two were cryptographically isolated from one another — as, indeed, they were intended to be. So, for example, imagine that some industrial control system uses a one-time secure pairing to communicate with its controlling monitor, but as a convenience allows for "Just Works" pairing to any nearby Smartphone for read-only passive monitoring. An entirely reasonable architectural design decision. But now, any of those nearby unauthorized Smartphones have the means to intercept, re-key and perform an active man-in-the-middle attack of the main control link that was assumed to be secure.

For those using the Bluetooth features of versions 4.0 through 5.0 there was no push or rush to implement the tighter restrictions in v5.1, if 5.1's additional functional enhancements where not needed. v5.0 was assumed to be secure. That all changed last week. We can expect to see a move to v5.1 for any "dual mode" Bluetooth devices.

And, as is always the case now, those Bluetooth devices that are not part of some actively maintained upgrade cycle will henceforth remain forever vulnerable to this newly revealed class of cross-transport attack.

https://www.kb.cert.org/vuls/id/589825/
https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/bluetooth-security/blurtooth/