Security Now! #728 - 08-20-19

The KNOB is Broken

This week on Security Now!

This week we look at last week's monthly patch Tuesday, and its collision with third-party A/V add-ons. We examine four year of Kaspersky unique-web-user tracking, look again at Tavis Ormandy's discovery of the secret undocumented "CTF" protocol, wondering WTF is CTF? We also note a new and devastating strategy in the ransomeware battle which hit Texas last Friday. We also have the sad demise of Extended Validation certificates, the further removal of FTP support from web browsers, Google's campaign to still-further reduce web certificate lifetimes, and Netflix's discovery of eight implementation flaws in the new HTTP/2 protocol. We'll cover a bit of miscellany, update on my file syncing journey, touch on SQRL news and SpinRite... and conclude with a look at the most recent attack on Bluetooth pairing negotiation which renders all Bluetooth associations vulnerable to a trivial attack.

Security News

Last Tuesday was another busy and important patch Tuesday
Of the 93 vulnerabilities Microsoft patched today, 29 are rated Critical and 64 are rated Important in severity.

Happily, for a change, none of the vulnerabilities patch last Tuesday were known to be under active attack nor had any details published publicly.

However, patching remained important since last Tuesday patched four remote code execution bugs in Microsoft's recently troubled Remote Desktop system. As we know, the previous wormable "BlueKeep" vulnerability affecting exposed Remote Desktop Servers was never "wormed" because, in my opinion, it is so trivial to exploit that adding worm "automation" didn't add any useful value. This is relevant since Simon Pope, the Director of Incident Response for the Microsoft Security Response Center (MSRC), blogged that two of the new four RDS bugs are also "wormable." Consequently, patching CVE-2019-1181 and CVE-2019-1182 should not be delayed for any enterprise of users who are running RDS. However, RDS is not RDP. RDS is Remote Desktop Services which was once called "Terminal Services" so it will primarily affect higher end entities which was running full remote access Terminal Services servers. Microsoft discovered these four new problems internally while refocusing upon RDP and RDS with an eye to tightening them up.

Beyond the four new problems found in RDS, there are seven RCEs impacting the Chakra scripting engine (which is present in Microsoft's Edge and other Microsoft apps, two RCEs in Microsoft Hyper-V VM technology, six RCEs in the Microsoft Graphics component, one in Outlook, two in Word, two in the Windows DHCP client, two in the older Scripting Engine component, and one in the VBScript engine.

And there is also a patch for a vulnerability in the CTF protocol that impacts all Windows versions since Windows XP.

So, overall, this month's August 2019 Patch Tuesday is both large and important. As I notes above, of the 93 vulnerabilities Microsoft patched, 29 are rated Critical and 64 are rated Important in severity.

Furthermore, with this occasion, Microsoft also wanted to remind users that Windows 7 and Windows Server 2008 R2 will be out of extended support and no longer receiving updates as of January 14, 2020.

Microsoft said: "*We strongly recommend that you update any computers running Windows 7 or Windows Server 2008 R2 so you will continue receiving security updates.*"

In other non-Microsoft security update news...

It's worth noting that Adobe, SAP, and VMWare have also published their respective security updates earlier today.

Adobe had fixes for Photoshop, Experience Manager, Acrobat/Reader, the Creative Cloud desktop app, Prelude, Premiere Pro, Character Animator, and After Effects. And, also notably, no Flash security updates this month.

And speaking of Patch Tuesday... 3rd-Party A/V Strikes Again!
Recall how Microsoft has decided to stop co-signing their updates with both SHA-1 and SHA-256, choosing to drop SHA-1 signing... That last month, July, was the last month of patches where both signatures would be present... And that only those systems which first updated with awareness of SHA-256 signing of security updates would be able to receive =ANY= starting with August.

Well, this is August, and things did indeed break, though not what Microsoft, or anyone else, was expecting. It turns out that the Symantec and Norton 3rd-party A/V tools have been making it their business to "protect" users from unsigned Windows updates... Which is kinda weird, since that's the whole reason they are signed in the first place... So that Windows can, will and does robustly protect itself from any and all possibility of accepting malicious updates.

So, apparently, the nosey Symantec and Norton A/V update protectors didn't get the memo about the signing changing on Windows updates. And apparently they were also both only checking the older SHA-1 signatures... Since last Tuesday, when the SHA-1 signatures were dropped from the updates, as planned for the past six months... Both of those A/V systems refused to allow their client machines to receive =ANY= of Microsoft's valid Patch Tuesday updates. Whoopsie!

There's little that Microsoft can do at this point, but Microsoft Update knows all about the inventories of the machines it's updating, so it has stopped attempting to send any updates to any Windows machines containing Symantec or Norton A/V ... and, as a consequence, in the meantime leaving those users MORE rather than less vulnerable to exploits.

https://support.microsoft.com/en-us/help/4512486/windows-7-update-kb4512486

Microsoft wrote under Known issues in this update:

> "Microsoft and Symantec have identified an issue that occurs when a device is running any Symantec or Norton antivirus program and installs updates for Windows that are signed with SHA-2 certificates only. The Windows updates are blocked or deleted by the antivirus program during installation, which may then cause Windows to stop working or fail to start." --
> "Microsoft has temporarily placed a safeguard hold on devices with an affected version of Symantec Antivirus or Norton Antivirus installed to prevent them from receiving this type of Windows update until a solution is available. We recommend that you do not manually install affected updates until a solution is available.
>
> Guidance for Symantec customers can be found in the Symantec support article.
> https://support.symantec.com/us/en/article.tech255857.html

Symantec: "Windows 7/Windows 2008 R2 updates that are only SHA-2 signed are not available with Symantec Endpoint Protection installed"...

> Symantec has identified the potential for a negative interaction between Symantec Endpoint Protection and the contents of future Windows Updates as a result of the changes in this Microsoft KB. Out of an abundance of caution, Symantec and Microsoft worked together to only allow the update to be visible to versions of Symantec Endpoint Protection that fully support SHA-2 signed Windows executables replaced by this and future updates to Windows 7 SP1 and Windows 2008 R2 SP1. Symantec is actively working on multiple releases of Symantec Endpoint Protection to address this situation. This document will be updated as each release becomes available for distribution and include details on how each update can be acquired.

So... If you are a Symantec or Norton user and are not already aware, you may want to keep an eye out for any of this which might affect you.

And, in another foul-up this month… Microsoft:

> "After installing this update, applications that were made using Visual Basic 6 (VB6), macros using Visual Basic for Applications (VBA), and scripts or apps using Visual Basic Scripting Edition (VBScript) may stop responding and you may receive an "invalid procedure call error. This issue is resolved in KB4517297, which is an optional update. It is now available on Microsoft Update Catalog and Windows Server Update Services (WSUS)."

Kaspersky facilitates independent web tracking:
And while we're on the topic of annoying things that A/V does to -- I mean for -- us, we have this bit of news from Kaspersky: Starting in late 2015, Kaspersky's A/V solutions -- both the free and paid editions -- have been injecting JavaScript into every web page displayed by their users. The JavaScript, which is placed into the page's DOM (Document Object Model) can therefore be readily seen by and parsed by any other script running on the page -- the hosting website and ads. What makes this an issue of concern is that the injection is being performed by the user's client side Kaspersky A/V and it contains a GUID -- a globally unique identifier -- which is unique to each Kaspersky user.

```
<script type="text/javascript" src="https://gc.kis.v2.scr.kaspersky-
labs.com/9344FDA7-AFDF-4BA0-A915-4D7EEB9A6615/main.js" charset="UTF-8">
</script>
```

Therefore, and the tech press has noted, anyone using Kaspersky A/V can be uniquely identified and tracked regardless of what other blocking or anti-tracking mechanism they might employ -- including Incognito mode, which this renders somewhat less private.

However, also note from the JavaScript above, that this script is loading a larger JavaScript blob from a Kaspersky Labs server with the GUID in the path. This means that the web browser of every Kaspersky user is phoning home to Kaspersky for every web page the user visits and identifying not only the user by GUID, but also providing Kaspersky with perfect user-tracking

information, since the script fetch will contain an HTTP Referer: header providing the exact location and page of every user fetch.

So that's the way things have been since last 2015. After the potential of this for abuse was brought to Kaspersky's attention and made public, they, last month, switched it from a unique per-user GUID to a unique per-Kaspersky-version GUID. So now, while Kaspersky themselves, sites visited and those sites' advertising syndicators will no longer have an immutable handle for tracking, they will at least know exactly which A/V and version every Kaspersky user is using.

And note that all of this is made possible because Kaspersky has planted a CA signing certificate into every user's machine and is silently intercepting and injecting its own code into every web page viewed.

Leo... I am very glad that neither of us have any of that crap happening on our systems.

So, what the heck is "CTF" ??

In the first item this week I mentioned in passing: "And there is also a patch for a vulnerability in the CTF protocol that impacts all Windows versions since Windows XP." The WHAT protocol? No, not WTF protocol... CTF protocol.

Once again, Google's Tavis Ormandy discovered this buggy protocol which, if hackers or malware had already gained a tentative foothold on a user's computer could use to take over any app, high-privileged applications, or the entire OS, as a whole. So it turns out that CTF -- which no one has ever heard of -- is a little-known Microsoft protocol used by all Windows operating systems since Windows XP. And, naturally, it's insecure and can be easily exploited.

What is CTF? No one knows for sure. We don't know what "CTF" stands for. Tavis was never able to determine what it means despite rummaging through all of Microsoft's documentation. So perhaps it really is "WTF."

What Tavis DID learn was that CTF is part of the Windows Text Services Framework (TSF), the system that manages the text shown inside Windows and Windows applications. When users start an app, Windows also starts a CTF client for that app. The CTF client receives instructions from a CTF server about the OS system language and the keyboard input methods.

If the OS input method changes from one language to another, then the CTF server notifies all CTF clients, who then change the language in each Windows app accordingly, in real-time. And Tavis quickly discovered to his shock and dismay that the communications between CTF clients and the CTF servers are not properly authenticated or secured. Or as Tavis put it: "There is no access control in CTF. Any application, any user - even sandboxed processes - can connect to any CTF session. Clients are expected to report their thread id, process id and HWND (Main Windows messaging Handle), but there is no authentication involved and you can simply lie."

Tavis added: "So you could connect to another user's active session and take over any application, or wait for an Administrator to login and compromise their session." An attacker that hijacks another app's CTF session can then send commands to that app, posing as the server -- normally expected to be the Windows OS.

Attackers can use this loophole to either steal data from other apps, or they can use it to issue commands in the name of those apps. If the apps run with high-privileges, then those actions allow the attacker to take full control over a victim's computer. And according to Tavis, any app or Windows process is up for grabs. Because of CTF's role -- to show text inside ANY app or service -- there's a CTF session for literally everything and every user interface element on a Windows OS.

To demonstrate the dangers Tavis recorded a demo in which he hijacked the CTF session of the Windows login screen -- which, as we know, Microsoft has made a huge deal about being highly privileged, isolated and sacrosanct. Thus Tavis easily demonstrated that everything in Windows is hackable because of CTF.

https://googleprojectzero.blogspot.com/2019/08/down-rabbit-hole.html

Tavis concludes his long and wonderfully detailed "Down the Rabbit Hole" blog posting but noting:

> I've implemented this attack in ctftool, follow the steps here to try it.
> https://github.com/taviso/ctftool#Exploit
>
> So what does it all mean?
>
> Even without bugs, the CTF protocol allows applications to exchange input and read each other's content. However, there are a lot of protocol bugs that allow taking complete control of almost any other application.
>
> It will be interesting to see how Microsoft decides to modernize the protocol.
>
> If you want to investigate further, I'm releasing the tool I developed for this project.
> https://github.com/taviso/ctftool
>
> Conclusion
>
> It took a lot of effort and research to reach the point that I could understand enough of CTF to realize it's broken. These are the kind of hidden attack surfaces where bugs last for years. It turns out it was possible to reach across sessions and violate NT security boundaries for nearly twenty years, and nobody noticed.

23 "Government Agencies" in Texas...

... were hit with a well-coordinated ransomware attack last Friday, August 16th.

23 Texas entities – the majority of which are local governments – were hit by a ransomware attack on Friday that Texas officials say is part of a targeted attack launched by a single threat actor. Details still remain scant about the specific agencies hit by the ransomware attacks, which began on the morning of Aug. 16, as well as which systems are impacted. However, the Texas Department of Information Resources (DIR) as of Saturday night did say that responders are

actively working with all 23 entities to bring their systems back online, and that the State of Texas systems and networks are not impacted. So what we DO know is that 23 agencies were knocked offline and presumably encrypted by the simultaneous attack.

The Texas Department of Information Resources website posted a statement saying that "Currently, DIR, the Texas Military Department, and the Texas A&M University System's Cyberresponse and Security Operations Center teams are deploying resources to the most critically impacted jurisdictions. Further resources will be deployed as they are requested."

When pressed for additional details the Texas DIR declined to elucidate any further stating "due to security concerns," and saying only that they were smaller, local governments."

The DIR also did not provide information about which systems are down, how systems were first infected, and the specific amount of ransom. In their reporting of this, Threatpost reached out to representatives from Dallas, Houston and Austin for comment on whether they were impacted by the attack. While representatives from Dallas and Austin have not yet responded, a spokesperson from Houston told Threatpost that "as far as we know, Houston has not been affected."

According to a statement sent to Threatpost: "The city of Houston is aware that a ransomware attack has affected several local government agencies throughout Texas. We are in contact with the Texas State Operations Center and will monitor the latest developments. The Mayor's Office of Homeland Security and the IT Services Department will continue to proactively work to secure and protect the city's assets."

However, the DIR said that at this time [not surprisingly, given what we know], evidence gathered indicates the attacks came from one single threat actor.

Allan Liska, threat intel analyst with Recorded Future, told Threatpost that the attacks signify an important shift in the ransomware attack model. Typically, state and local governments have been "targets of opportunity" for ransomware attacks – with the gangs behind Ryuk [Ree-Ook] and SamSam appearing to stumble onto previous state and local governments targets. However, this incident appears to be the first where a string of governments were actively being targeted in an attack.

Allan said: "This is the first time there's been an attack against several local governments in a state… this is big, it's a game-changer. This will change the model going forward [for attackers], and that will be a problem for governments."

Allan also noted that one advantage Texas has is that it has a consolidated incident response. The response team is centralized for cities and counties in emergencies. This makes it much easier when there is a problem like this to find and reach out to a main contact. There's someone to call.

I looked around for any update to the initial reporting and even the latest news from just hours ago is just repeating the news from last Friday. Everyone is being quiet. But they are now saying that it's 23 individual cities across Texas.

RIP, EV: The coming demise of Extended Validation (EV) certificates
Safari has already removed all EV Certificate company info from the address bar, most mobile browsers are not showing it, and now both Chrome and Firefox browsers for the desktop have announced that they will soon be removing the "EV" indication.

https://groups.google.com/a/chromium.org/forum/#!topic/security-dev/h1bTcoTpfeI

Chrome's Google Groups post was titled: "Upcoming Change to Chrome's Identity Indicators"

As part of a series of data-driven changes to Chrome's security indicators, the Chrome Security UX team is announcing a change to the Extended Validation (EV) certificate indicator on certain websites starting in Chrome 77. On HTTPS websites using EV certificates, Chrome currently displays an EV badge [containing the name of the EV certificate holder] to the left of the URL bar. Starting with Version 77, Chrome will move this UI indication down to the Page Info, which is accessed by clicking the lock icon. [In other words, effectively nullifying its impact.]

Through our own research as well as a survey of prior academic work, the Chrome Security UX team has determined that the EV UI does not protect users as intended. Users do not appear to make secure choices (such as not entering password or credit card information) when the UI is altered or removed, as would be necessary for EV UI to provide meaningful protection. Further, the EV badge takes up valuable screen real estate, can present actively confusing company names in prominent UI, and interferes with Chrome's product direction towards [a] neutral, rather than [a] positive, display for secure connections. [In other words, secure is intended to be the norm going forward and non-secured sites being denigrated.] Because of these problems and its limited utility, we believe it belongs better in Page Info.

Altering the EV UI is a part of a wider trend among browsers to improve their Security UI surfaces in light of recent advances in understanding of this problem space. In 2018, Apple announced a similar change to Safari that coincided with the release of iOS 12 and macOS 10.14 and has been implemented as such ever since.

And shortly following Chrome's announcement, Mozilla also announced that starting in Firefox 70 they will be removing the EV certificate's identity information from the address bar:

In desktop Firefox 70, we intend to remove Extended Validation (EV) indicators from the identity block (the left hand side of the URL bar which is used to display security / privacy information). We will add additional EV information to the identity panel instead, effectively reducing the exposure of EV information to users while keeping it easily accessible.

They wrote: The effectiveness of EV has been called into question numerous times over the last few years, there are serious doubts whether users notice the absence of positive security indicators and proof of concepts have been pitting EV against domains for phishing.

More recently, it has been shown that EV certificates with colliding entity names can be generated by choosing a different jurisdiction. 18 months have passed since then and no changes that address this problem have been identified.

The Chrome team recently removed EV indicators from the URL bar in Canary and announced their intent to ship this change in Chrome 77. Safari is also no longer showing the EV entity name instead of the domain name in their URL bar, distinguishing EV only by the green color. Edge is also no longer showing the EV entity name in their URL bar.

So... RIP EV

And... So long FTP!
Google Chrome Team's posting: Intent to Remove: Deprecate FTP support
https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/e1hkwUL4p3w/11sdjpuMAgAJ

Summary: Deprecate and remove support for FTP URLs.

> The current FTP implementation in Google Chrome has no support for encrypted connections (FTPS), nor proxies. Usage of FTP in the browser is sufficiently low that it is no longer viable to invest in improving the existing FTP client. In addition more capable FTP clients are available on all affected platforms.
>
> Google Chrome 72+ removed support for fetching document subresources over FTP and rendering of top level FTP resources. Currently navigating to FTP URLs result in showing a directory listing or a download depending on the type of resource. A bug in Google Chrome 74+ resulted in dropping support for accessing FTP URLs over HTTP proxies. Proxy support for FTP was removed entirely in Google Chrome 76.
>
> Remaining capabilities of Google Chrome's FTP implementation are restricted to either displaying a directory listing or downloading a resource over unencrypted connections. We would like to deprecate and remove this remaining functionality rather than maintain an insecure FTP implementation.

To that end, Chrome will soon be getting a new "DisableFTP" flag which will not be enabled at first. But this will allow Chrome to slowly but surely move away from supporting FTP. At some point it will be enabled by default so that only those who need to use it -- for some reason -- will still be able to. But any such individual should really switch to a separate full-blown FTP client.

Reducing Certificate Lifespans
Two months ago, in June, Google's Ryan Sleevi (who's a good guy, by the way), introduced a ballot measure proposing to reduce maximum browser security certificate lifetimes to essentially one year. The measure was titled "Ballot SCXX: Improve Certificate Lifetimes." Just the fact that Google said "Improve Certificate Lifetimes" when "Reduce Certificate Lifetimes" is what they really mean demonstrates that they clearly recognize that certificate revocation is broken.

My first thought upon learning of this was, "of course they are", since as longtime listeners of this podcast well know, Google's Chrome browser certificate revocation is **not** completely and utterly broken; rather, it is **non-existent** for all practical purposes. Chrome doesn't even BOTHER checking non-EV certificates with their CRLSET system. (And now they're working to kill

EV certificates, too!... the only thing they were checking.)

For some unfathomable reason back in the dawn of Chrome, someone must have made the decision that since certificate revocation was currently imperfect it was therefore of no value. So unlike every other browser on the planet, they would simply ignore it. It was an irresponsible and unconscionable choice. Oh, sure, they have their own web browser rigorously checking any and all of their **own** properties' certificates. You don't **dare** mess with a Google-derived cert. But they don't bother to check anyone else's. When I clearly demonstrated this five years ago, back in May of 2014, by creating a deliberately revoked certificate which Chrome gleefully accepted Google manually added that one certificate's signature to Chrome's short cert black list. So I created another which was then once again honored.

I'm sure that this, and I, annoyed them by bringing attention to this original sin of Chrome's. It wasn't my intention to annoy them. But I did hope that by bringing this to light we might see this fixed since back when Chrome was becoming more and more popular and influential. All of the other browsers were doing the right thing. And, again... yes... the existing system was imperfect. But it made more sense then -- and it still does now -- to fix it, rather than to simply ignore it.

I haven't revisited any of this since back then, so a little over 5 years ago. But anyone who is interested in that research and coverage can find those original pages at https://www.grc.com/revocation.htm

Back then, during that research I encountered the perfect solution to the problem, which is known as OCSP Stapling. OCSP is the "Online Certificate Status Protocol". Certificates contain the URL of their issuer's OCSP server -- which allows the current instantaneous status of the certificate to be verified. This enables web browsers to query certificate OCSP servers to receive a completely current up-to-the-instant check on the current validity of the certificate.

The trouble is, this has been an emerging standard, and in the beginning OCSP servers could not always be counted upon to reply quickly, if at all. This either slowed things down while browsers waited (something browsers really hate to do), or browsers were forced to take the position of trusting unless denied. But a perfect solution DOES and HAS existed for years: OCSP Stapling. When OCSP Stapling is used, the web server that's offering and asserting the validity of the certificate obtains and caches an updated and signed assertion of the =CURRENT= validity of the certificate and then "staples" it to the certificate which it offers to the web browser. The browser checks the signature of the certificate and of the recent OCSP validity assertion. Think of the power this gives CA's. Now they can revoke a certificate and it will become untrusted immediately… and with ZERO overhead introduced by browsers and no browser delay.

Two years ago, in July of 2017, Cloudflare's Nick Sullivan blogged with the title: "High-reliability OCSP stapling and why it matters": At Cloudflare our focus is making the internet faster and more secure. Today we are announcing a new enhancement to our HTTPS service: High-Reliability OCSP stapling. This feature is a step towards enabling an important security feature on the web: certificate revocation checking. Reliable OCSP stapling also improves connection times by up to 30% in some cases. In this post, we'll explore the importance of certificate revocation checking in HTTPS, the challenges involved in making it reliable, and how we built a robust OCSP stapling service.

https://blog.cloudflare.com/high-reliability-ocsp-stapling/

Today OCSP stapling is widely available. Windows, Apache and Nginx all support it, as do CDNs such as AWS and other web providers. So there's no longer any good reason NOT to simply make its support mandatory in the future.

https://www.digicert.com/enabling-ocsp-stapling.htm

Incredibly, Chrome doesn't perform **any** useful revocation checking. So, to minimize the exposure to a revoked certificate they want ALL certificates to die more rapidly through their natural self-expiration. But that still leaves us with a big mess -- and a gaping hole for exploitation. It feels as though Google is working to incrementally chip away at the Certificate Authority model, with the aim of eventually bringing us to the ACME protocol where we know NOTHING about a certificate -- where a certificate makes NO assertion -- other than the fact that one automated web server requested and received a domain-validating certificate from an ACME server.

That doesn't feel like the right future for the Internet. Moving forward, it seems so clear that we are going to need more than the assurance of encryption for our connections. We're going to really need to know **who** we're talking to at the other end. This is the vital service that Certificate Authorities have always provided. Are they perfect? No. But neither was OCSP, and that HAS been fixed -- completely -- with Stapling. There's just been too little pressure to implement it so far. Rather than discarding Certificate Authorities, and the vital service they provide, we should make that system stronger, too.

Someday the DANE protocol -- DNS-based Authentication of Named Entities -- might happen. But DNS needs to be made significantly stronger with DNSSEC before that can happen. The much more proximate solution is to move OCSP Stapling into the forefront by visually rewarding, in the browser's UI, those sites whose certificates carry a freshly stapled OCSP assertion. That would give sites an incentive to implement true zero-overhead certificate revocation checking.

HTTP/2 goes to the Movies

This is the second time we've reported that Netflix has performed some very nice security research.

HTTP/2 is =vastly= more complex than HTTP/1.1. We've talked about HTTP/2's many new features previously. In short, whereas a single HTTP/1.1 connection makes a request and waits for a reply, then might make another request during the same connection, or might terminate it and open up another... a single HTTP/2 is inherently a powerful multiplex connection supporting multiple simultaneous overlapping streams supporting multiple simultaneous outstanding requests, differing stream priorities and even the ability for the server to anticipate future not-yet-requested assets and send them ahead, anyway. BOY does it feel over-engineered. But it's what today's massive web pages require.

The trouble is... all of those features, many which are not yet even being used, are also extremely tricky to implement. So Netflix took a look at a number of actual real-world implementations and found EIGHT ways to clog-up the pipes and bring a burly web server to its knees with just a few carefully-chosen packets.

Here's what Netflix wrote:

Overview:

Netflix has discovered several resource exhaustion vectors affecting a variety of third-party HTTP/2 implementations. These attack vectors can be used to launch DoS attacks against servers that support HTTP/2 communication.

Netflix worked with Google and CERT/CC to coordinate disclosure to the Internet community.

Today, a number of vendors have announced patches to correct this suboptimal behavior. While we haven't detected these vulnerabilities in our open source packages, we are issuing this security advisory to document our findings and to further assist the Internet security community in remediating these issues.
Impact:

There are three broad areas of information security: confidentiality (information can't be read by unauthorized people), integrity (information can't be changed by unauthorized people), and availability (information and systems are available when you want them). All of the changes announced today are in the "availability" category. These HTTP/2 vulnerabilities do not allow an attacker to leak or modify information.

Rather, they allow a small number of low bandwidth malicious sessions to prevent connection participants from doing additional work. These attacks are likely to exhaust resources such that other connections or processes on the same machine may also be impacted or crash.
The Weaknesses:

HTTP/2 (defined in RFCs 7540 and 7541) represents a significant change from HTTP/1.1. There are several new capabilities, including header compression and multiplexing of data from multiple streams, which make this attractive to the user community. To support these new features, HTTP/2 has grown to encompass some of the complexity of a Layer 3 transport protocol:

- Data is now carried in binary frames;
- There are both per-connection and per-stream windows that define how much data can be sent;
- There are several ICMP-like control messages (ping, reset, and settings frames, for example) which operate at the HTTP/2 connection layer; and,
- This is a fairly robust concept of stream prioritization.

What did they find? The description of the eight CVE's will give us enough of a feel for it:

Many of the attack vectors we found (and which were fixed today) are variants on a theme: a malicious client asks the server to do something which generates a response, but the client refuses to read the response. This exercises the server's queue management code. Depending on how the server handles its queues, the client can force it to consume excess memory and CPU while processing its requests.

These are the attacks which are being disclosed today, all discovered by Jonathan Looney of Netflix, except for CVE-2019-9518 which was discovered by Google:

- CVE-2019-9511 "Data Dribble": The attacker requests a large amount of data from a specified resource over multiple streams. They manipulate window size and stream priority to force the server to queue the data in 1-byte chunks. Depending on how efficiently this data is queued, this can consume excess CPU, memory, or both, potentially leading to a denial of service.

- CVE-2019-9512 "Ping Flood": The attacker sends continual pings to an HTTP/2 peer, causing the peer to build an internal queue of responses. Depending on how efficiently this data is queued, this can consume excess CPU, memory, or both, potentially leading to a denial of service.

- CVE-2019-9513 "Resource Loop": The attacker creates multiple request streams and continually shuffles the priority of the streams in a way that causes substantial churn to the priority tree. This can consume excess CPU, potentially leading to a denial of service.

- CVE-2019-9514 "Reset Flood": The attacker opens a number of streams and sends an invalid request over each stream that should solicit a stream of RST_STREAM frames from the peer. Depending on how the peer queues the RST_STREAM frames, this can consume excess memory, CPU, or both, potentially leading to a denial of service.

- CVE-2019-9515 "Settings Flood": The attacker sends a stream of SETTINGS frames to the peer. Since the RFC requires that the peer reply with one acknowledgement per SETTINGS frame, an empty SETTINGS frame is almost equivalent in behavior to a ping. Depending on how efficiently this data is queued, this can consume excess CPU, memory, or both, potentially leading to a denial of service.

- CVE-2019-9516 "0-Length Headers Leak": The attacker sends a stream of headers with a 0-length header name and 0-length header value, optionally Huffman encoded into 1-byte or greater headers. Some implementations allocate memory for these headers and keep the allocation alive until the session dies. This can consume excess memory, potentially leading to a denial of service.

- CVE-2019-9517 "Internal Data Buffering": The attacker opens the HTTP/2 window so the peer can send without constraint; however, they leave the TCP window closed so the peer cannot actually write (many of) the bytes on the wire. The attacker then sends a stream of requests for a large response object. Depending on how the servers queue the responses, this can consume excess memory, CPU, or both, potentially leading to a denial of service.

- CVE-2019-9518 “Empty Frames Flood”: The attacker sends a stream of frames with an empty payload and without the end-of-stream flag. These frames can be DATA, HEADERS, CONTINUATION and/or PUSH_PROMISE. The peer spends time processing each frame disproportionate to attack bandwidth. This can consume excess CPU, potentially leading to a denial of service. (Discovered by Piotr Sikora of Google)

Workarounds and Fixes:

In most cases, an immediate workaround is to disable HTTP/2 support.

Miscellany

David Theese @davidtheese
Traveling from Phoenix on Thursday to hear your SQRL talk. Looking forward to meeting you! Long-time enjoyer of SN!

https://www.meetup.com/OWASP-OC/events/263576551/
https://www.grc.com/calendar.htm

File Sync Update

Security Now Episode #734 will be "Steve's File Sync Conclusions"

SpinRite

The second anniversary of my first date with Lorrie is approaching, so I went looking for the exact date from our early eMail correspondence. I found it, and right next to it was this note from a SpinRite user that was one of those heart warmers that makes me so glad this product exists:

To: Steve
From: Yann Fitzmorris
Subject: another success story SpinRite data recovery

Dear Steve and GRC team,

I purchased SpinRite a few years ago and have been using it to keep my drives in good health. I personally have never had to use it for data recovery, however, a friend asked for my help this week because her laptop would no longer boot to the log in screen. Her laptop contained the only copy of pictures and videos of the first 2 years of her daughter's life.

It wasn't looking good for the patient - when I plugged the drive into an external dock, no OS would recognize the drive.

I used my dedicated PC for SpinRite, plugged in the drive and ran level 2.

Success!

We plugged the drive into the doc and we were able to recover ALL pictures and videos - over 70GB. Needless to say, my friend will now seriously consider a backup solution, and I'm hoping she will buy her own copy of SpinRite to show her gratitude for this amazing product!

Thanks again for all your hard work and research. Love the Security Now podcast as well - I've been a listener since 2015.

regards,
Yann Fitzmorris
twitter: @yfitzmorris

The KNOB is Broken

The paper was included in the Proceedings of the 28th USENIX Security Symposium last week in Santa Clara, California. The paper's full title is:

"The KNOB is Broken: Exploiting Low Entropy in the Encryption Key Negotiation Of Bluetooth BR/EDR"

Three security researchers reported on their analysis of more than 14 Bluetooth chips from different vendors: https://www.usenix.org/system/files/sec19-antonioli.pdf

First of all, "Bluetooth BR/EDR" stands for Basic Rate/Enhanced Data Rate and is the original Bluetooth, sometimes referred to as "Bluetooth Classic" and present in more than a billion Bluetooth-enabled devices.

Abstract:
We present an attack on the encryption key negotiation protocol of Bluetooth BR/EDR. The attack allows a third party, without knowledge of any secret material (such as link and encryption keys), to make two (or more) victims agree on an encryption key with only 1 byte (8 bits) of entropy. Such low entropy enables the attacker to easily brute force the negotiated encryption keys, decrypt the eavesdropped ciphertext, and inject valid encrypted messages (in real-time). The attack is stealthy because the encryption key negotiation is transparent to the Bluetooth users. The attack is standard-compliant because all Bluetooth BR/EDR versions are required to support encryption keys with entropy between 1 and 16 bytes and do not secure the key negotiation protocol. As a result, the attacker completely breaks Bluetooth BR/EDR security without being detected. We call our attack "Key Negotiation Of Bluetooth" (KNOB) attack. The attack targets the firmware of the Bluetooth chip because the firmware (Bluetooth controller) implements all the security features of Bluetooth BR/EDR. As a standard-compliant attack, it is expected to be effective on any firmware that follows the specification and on any device using a vulnerable firmware. We describe how to perform the KNOB attack, and we implement it. We evaluate our implementation on more than 14 Bluetooth chips from popular manufacturers such as Intel, Broadcom, Apple, and Qualcomm. Our results demonstrate that all tested devices are vulnerable to the KNOB attack. We discuss countermeasures to fix the Bluetooth specification and its implementation.

CERT's vulnerability notice explains this obvious problem this way:
(Using Alice & Bob as the parties wishing to communicate securely and Charlie in the role of attacker.)

To establish an encrypted connection, two Bluetooth devices must pair with each other and establish a link key that is used to generate the encryption key. For example, assume that there are two controllers attempting to establish a connection: Alice and Bob. After authenticating the link key, Alice proposes that she and Bob use 16 bytes of entropy. This number, N, could be between 1 and 16 bytes. Bob can either accept this, reject this and abort the negotiation, or propose a smaller value. Bob may wish to propose a smaller N value because he (the controller) does not support the larger number of bytes proposed by Alice. After proposing a smaller

amount, Alice can accept it and request to activate link-layer encryption with Bob, which Bob can accept.

An attacker, Charlie, could force Alice and Bob to use a smaller N by intercepting Alice's proposal request to Bob and changing N. Charlie could lower N to as low as 1 byte, which Bob would subsequently accept since Bob supports 1 byte of entropy and it is within the range of the compliant values. Charlie could then intercept Bob's acceptance message to Alice and change the entropy proposal to 1 byte, which Alice would likely accept, because she may believe that Bob cannot support a larger N. Thus, both Alice and Bob would accept N and inform the Bluetooth hosts that encryption is active, without acknowledging or realizing that N is lower than either of them initially intended it to be.

So what we have is another classic cryptographic security downgrade attack similar to what we have run into many times through the years. It's amazing that Bluetooth is this mature and that we're only now noticing this oversight.

Under "Impact" CERT notes:
An unauthenticated, adjacent attacker can force two Bluetooth devices to use as low as 1 byte of entropy. This would make it easy for an attacker to brute force as it reduces the total number of possible keys to try, and would give them the ability to decrypt all of the traffic between the devices during that session.

And here's the really nutty part: The researchers note the following about half way through their 16-page paper:

> We do not see any reason to include the encryption key negotiation protocol in the specification of Bluetooth. From our experiments (presented in Section 5) we observe that if two devices are not attacked they **always** use it in the same way (a device proposes 16 bytes of entropy and the other accepts). Furthermore, the entropy reduction does not improve runtime performance because the size of the encryption key is fixed to 16 bytes even when its entropy is reduced.

So a bunch of Bluetooth stacks are now busily being updated including: Microsoft for Windows, Cisco for IP Phones and Webex, Google for Android, Apple for macOS, iOS, and watchOS, BlackBerry and, I'm sure, other.

Our longtime listeners will recall that I have several times observed that there is a large though brief period of inherent vulnerability during Bluetooth pairing. You have two unauthenticated devices hoping to perform a secure negotiation. It's simply not possible to do that securely without some covert out-of-band channel. It's just not. So I recommended that if someone really really needed Bluetooth security that they should stand out in the middle of a completely deserted parking lot to perform the pairing… and hope that no one is aiming high-gain antennas at them!