Security Now! #601 - 02-28-17

The First SHA-1 Collision

This week on Security Now!

If it leads it CloudBleeds; another project zero 90-day timer expires for Microsoft; this week's IoT head-shaker; a New York airport exposes critical server data for a year; another danger created by inline third party TLS-intercepting "middleboxes"; more judicial thrashing over fingerprint warrants; Amazon says no to Echo data warrant; a fun drone-enabled proof on concept is widely misunderstood; another example of A/V attack surface expansion; some additional Crypto education pointers and miscellany... and, finally, what does Google's deliberate creation of two SHA-1-colliding files actually mean?

Security News

The official name: Cloudflare Reverse Proxies are Dumping Uninitialized Memory

- Opened by Google's Tavis Ormandy on February 19th as part of Project zero.
- 90-day disclosure clock begins ticking.
- Issue fixed, resolved and closed four days later, on February 23rd.

- The Tick-Tock Breakdown:
  - https://bugs.chromium.org/p/project-zero/issues/detail?id=1139

- Tavis' first note:
  "(It took every ounce of strength not to call this issue "cloudbleed")"

  On February 17th 2017, I was working on a corpus distillation project, when I encountered some data that didn't match what I had been expecting. It's not unusual to find garbage, corrupt data, mislabeled data or just crazy non-conforming data...but the format of the data this time was confusing enough that I spent some time trying to debug what had gone wrong, wondering if it was a bug in my code. In fact, the data was bizarre enough that some colleagues around the Project Zero office even got intrigued.

  It became clear after a while we were looking at chunks of uninitialized memory interspersed with valid data. The program that this uninitialized data was coming from just happened to have the data I wanted in memory at the time. That solved the mystery, but some of the nearby memory had strings and objects that really seemed like they could be from a reverse proxy operated by cloudflare - a major cdn service.

  A while later, we figured out how to reproduce the problem. It looked like that if an html page hosted behind cloudflare had a specific combination of unbalanced tags, the proxy would intersperse pages of uninitialized memory into the output (kinda like heartbleed, but cloudflare specific and worse for reasons I'll explain later). My working theory was that this was related to their "ScrapeShield" feature which parses and obfuscates html - but because reverse proxies are shared between customers, it would affect *all* Cloudflare customers.

  We fetched a few live samples, and we observed encryption keys, cookies, passwords, chunks of POST data and even HTTPS requests for other major cloudflare-hosted sites from other users. Once we understood what we were seeing and the implications, we immediately stopped and contacted cloudflare security.

  This situation was unusual, PII was actively being downloaded by crawlers and users during normal usage, they just didn't understand what they were seeing. Seconds mattered here, emails to support on a friday evening were not going to cut it. I don't have any cloudflare contacts, so reached out for an urgent contact on twitter, and quickly reached the right people.

- "Could someone from cloudflare security urgently contact me."
  - https://twitter.com/taviso/status/832744397800214528

- After I explained the situation, cloudflare quickly reproduced the problem, told me they had convened an incident and had an initial mitigation in place within an hour.

- "You definitely got the right people. We have killed the affected services"

- Tavis' first follow-up:
  I worked with cloudflare over the weekend to help clean up where I could. I've verified that the original reproduction steps I sent cloudflare no longer work.

  We're discussing some remaining issues and Cloudflare are still working on their investigation. From our data, we believe the issue has been present for some time, I've given cloudflare all the information we have.

  This is an unusual type of vulnerability disclosure for Project Zero, and due to the third party nature of the data involved I'm reluctant to be as transparent as we normally are on our issue tracker. We've destroyed the samples we collected during analysis.

  Cloudflare have assured me they will prepare a detailed postmortem for their customers once the issue is resolved. They have an excellent reputation for transparency, so that's good enough for me.

- Tavis' next follow-up:
  We've been trying to help clean up cached pages inadvertently crawled at Google. This is just a bandaid, but we're doing what we can. Cloudflare customers are going to need to decide if they need to rotate secrets and notify their users based on the facts we know.

  I don't know if this issue was noticed and exploited, but I'm sure other crawlers have collected data and that users have saved or cached content and don't realize what they have, etc. We've discovered (and purged) cached pages that contain private messages from well-known services, PII from major sites that use cloudflare, and even plaintext API requests from a popular password manager that were sent over https (!!).

  Really impressed with Cloudflare's quick response, and how dedicated they are to cleaning up from this unfortunate issue. I told them I hoped they were able to get some sleep during the incident, and an engineer responded:

  “Its going to be a long weekend for us, but this is really important to us. We need to protect as many people as we can and we need to make sure this can never happen again on our watch.”

- Next from Tavis:
  We keep finding more sensitive data that we need to cleanup. I didn't realize how much of the internet was sitting behind a Cloudflare CDN until this incident.

  The examples we're finding are so bad, I cancelled some weekend plans to go into the office on Sunday to help build some tools to cleanup. I've informed cloudflare what I'm working on. I'm finding private messages from major dating sites, full messages from a well-known chat service, online password manager data, frames from adult video sites, hotel bookings. We're talking full https requests, client IP addresses, full responses, cookies, passwords, keys, data, everything.

  My current theory is that they had some code in their "ScrapeShield" feature that [paraphrasing for the podcast] wasn't checking if the HTML parsers returned a negative value (thus also a large positive value) because of malformed HTML. This would explain the data I'm seeing.

- Continuing from Travis:
  I had a call with Cloudflare, they reassured me they're planning on complete transparency and believe they can have a customer notification ready this week.

  I'm satisfied cloudflare are committed to doing the right thing, they've explained their current plan for disclosure and their rationale.

  We're still working on identifying data that needs to be purged from caches. Here's a heavily redacted random example of the data we're finding and purging, many major internet companies are using cloudflare.

  (Tavis shares a redacted snapshot of typical data.)

  Update from Cloudflare, they're confident they can get their notification ready by EOD Tuesday (Today) or early Wednesday.

  We're cleaning up some last remaining artifacts.

  Processing more removals this morning, and some further discussion on disclosure.

  (Here's another random redacted example from today.)
    - (Tavis is becoming more and more horrified by what he's finding.)

- We're now at February 22nd:
  Cloudflare told me that they couldn't make Tuesday due to more data they found that needs to be purged.

  They then told me Wednesday, but in a later reply started saying Thursday.

  I asked for a draft of their announcement, but they seemed evasive about it and clearly didn't want to do that. I'm really hoping they're not planning to downplay this. If the date keeps extending, they'll reach our "7-day" policy for actively exploited attacks.

https://security.googleblog.com/2013/05/disclosure-timeline-for-vulnerabilities.html

If an acceptable notification is not released on Thursday, we'll decide how we want to proceed.

Cloudflare sent a list of sites with more PII, but the list contained many well-known domains that were not using cloudflare (e.g. blogspot).

I let them know, and then tried to manually clean their list of obviously invalid requests to avoid any further delays. I filtered the remaining requests through a quick script I wrote to verify they were reasonable, which removed another 20 incorrect entries.

I'm sure this was a good faith mistake from a team rushing to protect users before publication today.

- Feb 23rd:
  I had a call with cloudflare, and explained that I was baffled why they were not sharing their notification with me.

  They gave several excuses that didn't make sense, then asked to speak to me on the phone to explain. They assured me it was on the way and they just needed my PGP key. I provided it to them, then heard no further response.

  I sent the following mail:
  - Thanks for the call John, I still haven't received anything.

    Can we commit to getting this out the door today? We're discussing whether we need to apply our "7-day" policy for actively exploited attacks, and while we were hoping the notification was going to be smooth sailing we don't know how to interpret the lack of cooperation here. I've been working with Marc this morning on more cleaning due to some incorrect data you sent us, I'm confident this will be done within hours.

    Let's commit to getting this out by 2PM PST.

    Tavis.

- Then Tavis posts:
  Here is another sample from the data we cleaned today, a private message from a popular dating site. Sigh. (Redacted message from "OKCupid".)

- And…
  Cloudflare pointed out their bug bounty program, but I noticed it has a top-tier reward of a t-shirt.

  https://hackerone.com/cloudflare

  Needless to say, this did not convey to me that they take the program seriously.

- And…
  Cloudflare explained that they pushed a change to production that logged malformed pages that were requested, and then sent me the list of URLs to double check.

  Many of the logged urls contained query strings from https requests that I don't think they intended to share. Nevertheless, we will keep the data safe and destroy it once we've checked if they match any cached pages.

  I can already see that the (huge) list does not contain many pages that still have cached content from before the bug was fixed.

- And…
  Cloudflare did finally send me a draft. It contains an excellent postmortem, but severely downplays the risk to customers.

  They've left it too late to negotiate on the content of the notification.

  Let's hope that their notification in combination with the details from this issue will be adequate explanation of what happened. I think we're waiting for cached links to start expiring, and then we're publishing whether they're ready or not.

- **Feb 23rd, Tavis sets the status to "Fixed" and notes:**
  OK, this is as done as it's going to be.

- **CloudFlare Responds**
  Nick Sullivan (@grittygrease) 2/23/17, 3:34 PM
  - Cloudflare had a bug. A bad one. In the spirit of transparency we've shared all the details.
  - https://blog.cloudflare.com/incident-report-on-memory-leak-caused-by-cloudflare-parser-bug/

- Incident report on memory leak caused by Cloudflare parser bug
  Feb 23rd by John Graham-Cumming
  https://blog.cloudflare.com/incident-report-on-memory-leak-caused-by-cloudflare-parser-bug/

  Last Friday, Tavis Ormandy from Google’s Project Zero contacted Cloudflare to report a security problem with our edge servers. He was seeing corrupted web pages being returned by some HTTP requests run through Cloudflare.

  It turned out that in some unusual circumstances, which I’ll detail below, our edge servers were running past the end of a buffer and returning memory that contained private information such as HTTP cookies, authentication tokens, HTTP POST bodies, and other sensitive data. And some of that data had been cached by search engines.

For the avoidance of doubt, Cloudflare customer SSL private keys were not leaked. Cloudflare has always terminated SSL connections through an isolated instance of NGINX that was not affected by this bug.

We quickly identified the problem and turned off three minor Cloudflare features (email obfuscation, Server-side Excludes and Automatic HTTPS Rewrites) that were all using the same HTML parser chain that was causing the leakage. At that point it was no longer possible for memory to be returned in an HTTP response.

Because of the seriousness of such a bug, a cross-functional team from software engineering, infosec and operations formed in San Francisco and London to fully understand the underlying cause, to understand the effect of the memory leakage, and to work with Google and other search engines to remove any cached HTTP responses.

Having a global team meant that, at 12 hour intervals, work was handed over between offices enabling staff to work on the problem 24 hours a day. The team has worked continuously to ensure that this bug and its consequences are fully dealt with. One of the advantages of being a service is that bugs can go from reported to fixed in minutes to hours instead of months. The industry standard time allowed to deploy a fix for a bug like this is usually three months; we were completely finished globally in under 7 hours with an initial mitigation in 47 minutes.

The bug was serious because the leaked memory could contain private information and because it had been cached by search engines. We have also not discovered any evidence of malicious exploits of the bug or other reports of its existence.

The greatest period of impact was from February 13 and February 18 with around 1 in every 3,300,000 HTTP requests through Cloudflare potentially resulting in memory leakage (that’s about 0.00003% of requests).

We are grateful that it was found by one of the world’s top security research teams and reported to us.

This blog post is rather long but, as is our tradition, we prefer to be open and technically detailed about problems that occur with our service.

The crux of the problem is that after decrypting the user's data, they parse into the page to insert and remove various things. Converting HTTP to HTTPS, etc. Sometimes removing privacy compromising and other unwanted page components.

- And... if the page they are parsing contained a specific type of HTML error, the parsing (another example of an interpreter failing catastrophically) would create a buffer overrun.

The Ragel code is converted into generated C code which is then compiled. The C code uses, in the classic C manner, pointers to the HTML document being parsed, and Ragel itself gives the user a lot of control of the movement of those pointers. The underlying bug occurs because of a pointer error.

-
  ```
  /* generated code */
  if ( ++p == pe )
    goto _test_eof;
  ```

- The root cause of the bug was that reaching the end of a buffer was checked using the equality operator and a pointer was able to step past the end of the buffer. This is known as a buffer overrun. Had the check been done using >= instead of == jumping over the buffer end would have been caught. The equality check is generated automatically by Ragel and was not part of the code that we wrote. This indicated that we were not using Ragel correctly.

  The Ragel code we wrote contained a bug that caused the pointer to jump over the end of the buffer and past the ability of an equality check to spot the buffer overrun.

- **The Internet Press Responds**
  - Google Just Discovered A Massive Web Leak... And You Might Want To Change All Your Passwords
    - https://www.forbes.com/sites/thomasbrewster/2017/02/24/google-just-discovered-a-massive-web-leak-and-you-might-want-to-change-all-your-passwords/#51de60ba3ca3

  - How to secure your data after the Cloudflare leak
    - https://flipboard.com/@flipboard/flip.it%2FlzUT8l-how-to-secure-your-data-after-the-cloud/f-0da2933867%2Ftechcrunch.com

  - Change Your Passwords. Now.
    - http://gizmodo.com/cloudbleed-password-memory-leak-cloudflare-1792709635

  - Cloudflare Bug Exposes Data Of Hundreds Of Thousands Of Customers
    - http://www.valuewalk.com/2017/02/cloudflare-bug-exposes-data-of-hundreds-of-thousands-of-customers/

  - Serious Cloudflare bug exposed a potpourri of secret customer data
    - https://arstechnica.com/security/2017/02/serious-cloudflare-bug-exposed-a-potpourri-of-secret-customer-data/

  - Passwords and dating site messages leaked by internet giant Cloudflare
    - http://www.theverge.com/2017/2/24/14723184/cloudflare-leak-cloudbleed-passwords-dating-site-messages

- Does it use CloudFlare?
  - http://www.doesitusecloudflare.com/

- The Wikipedia Page
  - https://en.wikipedia.org/wiki/Cloudbleed

- If you DO NEED to use a third-party TLS proxying service, don't use some random service no one has ever heard of. Use Cloudflare.

Google reports high-severity bug in Edge/IE, no patch available
Microsoft Edge and IE: Type confusion in HandleColumnBreakOnColumnSpanningElement
https://bugs.chromium.org/p/project-zero/issues/detail?id=1011

Tick Tock:

- Microsoft notified more than 90 days ago on November 25th.

- This is a readily exploitable crash affecting Edge and IE.

- Google's Ivan Fratric discovered, posted, and details:

  MSHTML!Layout::Patchable<Layout::PatchableArrayData<Layout::MultiColumnBox::SMultiColumnBoxItem> >::Readable is called which sets up rax.

  rcx is supposed to point to another object type, but in the PoC it points to an array of 32-bit integers allocated in Array<Math::SLayoutMeasure>::Create. This array stores offsets of table columns, and the values can be controlled by an attacker (with some limitations).

  On 00007ffe`8f330a59 the crash occurs because rax points to uninitialized memory.

  However, an attacker can affect rax by modifying table properties such as border-spacing and the width of the first element. Let's see what happens if an attacker can point rax to the memory he/she controls.

  Assuming an attacker can pass a check on line 00007ffe`8f330a59, MSHTML!Layout::Patchable<Layout::PatchableArrayData<Layout::MultiColumnBox::SMultiColumnBoxItem> >::Readable is called again with the same arguments. After that, through a series of dereferences starting from rax, a function pointer is obtained and stored in rdi.

  A CFG check is made on that function pointer and, assuming it passes, the attacker-controlled function pointer is called on line 00007ffe`8f330a80.

  Four days ago, on February 23rd, the 90-day disclosure timer expired.

  This surprising and worrisome disclosure of a still-unpatched bug generated some buzz and follow-up questions.

  Yesterday, he wrote:

- 1) I will not make any further comments on exploitability, at least not until the bug is fixed. The report has too much info on that as it is (I really didn't expect this one to miss the deadline).

- 2) The first step would be to determine why the type confusion occurred in the first place. Adding a type check somewhere in the vulnerable function might be sufficient, but it also might be just fixing the symptom and not the root cause. My hypothesis, given that there are 2 types of columns in DOM: html table columns and CSS columns, is that IE/Edge gets confused between the two.

- And... the US National Vulnerability Database has assigned this CVE-2017-0037, stating that: It "allows remote attackers to execute arbitrary code via vectors involving a crafted Cascading Style Sheets (CSS) token sequence and crafted JavaScript code that operates on a [table-header] element."

This week's IoT head-shaker:

- What are CloudPets?
  The idea is that friends or relatives can use the CloudPets smartphone app to record an audio message and send it to app on the parents' phone. The app then uploads the audio to the toy using Bluetooth LE. When the child presses the animal's right paw, it will play back the message. They can then record their own message by pressing the toy's other paw. The app then retrieves this audio reply via Bluetooth and sends it back to the friend-or-relative.

- What could possibly go wrong?

- Motherboard reports:
  https://motherboard.vice.com/en_us/article/internet-of-things-teddy-bear-leaked-2-million-parent-and-kids-message-recordings
  A company that sells internet-connected teddy bears that allow kids and their far-away parents to exchange heartfelt messages left more than 800,000 customer credentials, as well as two million message recordings, totally exposed online for anyone to see and listen.

  Since Christmas day of last year and at least until the first week of January, Spiral Toys left customer data of its CloudPets brand on a database that wasn't behind a firewall or password-protected. The MongoDB was easy to find using Shodan, a search engine makes it easy to find unprotected websites and servers, according to several security researchers who found and inspected the data.

  The exposed data included more than 800,000 emails and passwords, which are secured with the strong, and thus supposedly harder to crack, hashing function bcrypt. Unfortunately, however, a large number of these passwords were so weak that it's possible to crack them, according to Troy Hunt, a security researcher who maintains "Have I Been Pwned" and has analyzed the CloudPets data.

  Data from connected CloudPets teddy bears leaked and ransomed, exposing kids' voice messages
  - https://www.troyhunt.com/data-from-connected-cloudpets-teddy-bears-leaked-and-ransomed-exposing-kids-voice-messages/

> This morning, Paul Stone a researcher with the UK-based security firm Context, posted "Hacking Unicorns with Web Bluetooth" and revealed that the CloudPets' toys don't use any standard Bluetooth security features such as pairing encryption, when communicating back to their owner's smartphone's app. Anyone within range, Stone said, can connect to the toy, upload a message to the toy, "silently" trigger the toy's recording functionality, and "download the audio that the toy has recorded.
>
> So if you have a smartphone with Bluetooth, Stone explains, you can just connect to it and start sending audio messages to it. You don't even need to be within 10 meters (approximately 32 feet) if you use a directional antenna.
>
> Someone standing outside your house could easily connect to the toy, upload audio recordings, and receive audio from the microphone.

- Editorializing: Just imagine if or when, someday, our lives are FILLED with this sort of crap? Every day will be a new adventure! :-/

Security lapse exposed New York airport's critical servers for a year
http://www.zdnet.com/article/unsecured-servers-at-new-york-airport-left-exposed-for-a-year/
Zach Whittaker, reporting in his Zero Day column writes: The files included gigabytes of emails, sensitive government files, and a password list, which researchers say could give hackers "full access" to the airport's systems.

Stewart International Airport, 60 miles north of Manhattan, serves hundreds of thousands of passengers each year, and is regularly used by the military. The airport is known for accommodating charter flights of high-profile guests, including foreign dignitaries.

A security lapse left its server backups publicly exposed on the Internet since April of last year. The internet-connected storage drive contained several backup images of servers used by Stewart International Airport, but neither the backup drive nor the disk images were password protected, allowing anyone to access their contents.

Chris Vickery, lead security researcher of the MacKeeper Security Center, who helped to analyze the exposed data said the drive was "in essence, acting as a public web server" because the airport was backing up unprotected copies of its systems to a Buffalo-brand drive, installed by a contract third-party IT specialist.

Although the listing still appears on Shodan, the search engine for unprotected devices and databases, the drive has since been secured.

The files included 11 disk images, accounting for hundreds of gigabytes of files and folders, which when mounted included dozens of airport staff email accounts, sensitive human resources files, interoffice memos, payroll data, and what appears to be a large financial tracking database.

Many of the files we reviewed include "confidential" internal airport documents, which contain schematics and details of other core infrastructure.

Others belonging to Homeland Security agencies were marked "sensitive" but not classified, including comprehensive security plans, screening protocols, and arrival procedures for private jet passengers.

But one file contained a list of usernames and passwords for various devices and systems, allowing unfettered access to the airport's internal network, according to two security researchers.

Khalil Sehnaoui, founder of Krypton Security, and Brad "Renderman" Haines, a hacker and security researcher, analyzed the password file and a network schematic found among the files to determine the reach of a potential attacker.

They noted: "The password file would give us full access to every component of the internal network."

He added that the passwords in part relate to the airport's passenger processing system, provided by AirIT, which allows airport staff to manage passenger records, gates, and boarding.

That could allow a hacker to manipulate boarding passes and other passenger information.

Offering an example, they added: "For the best case scenario part, where no one really gets hurt, you could upgrade yourself to first class or just issue yourself a boarding pass to any destination served by departing planes. But in the wrong hands, it could also be used to issue valid boarding passes to people on the "no-fly" list, a government watchlist that prevents possible terrorists from boarding flights."

In more TLS-interception news: BlueCoat and other proxies hang up during TLS 1.3
Jay H Lee: https://bugs.chromium.org/p/chromium/issues/detail?id=694593

Chrome version 56 attempted to support TLS v1.3

What steps will reproduce the problem?
(1) BlueCoat 6.5 proxy.
(2) Chrome OS 56 or Chrome browser 56
(3) Attempt to connect to a Google service (youtube, accounts.google.com, etc.)

What is the expected result?
Successful connection. Client and proxy may negotiate down to TLS 1.2 instead of TLS 1.3.

What happens instead?
When Chrome attempts to connect via TLS 1.3, BlueCoat hangs up connection.

Further details:
We have at least one very large customer seeing similar issues against BlueCoat. The connection fails with SSL_HANDSHAKE_ERROR / ERR_CONNECTION_CLOSED. Customer found that restricting to TLS 1.2 via policy resolves the issue for Chrome 56 stable.

Other large EDU customers are seeing similar issues and we're working to gather details from them on proxy / firewall in use. Suspect many are using SSL / TLS inspection which is common among EDUs.

Marking this as ReleaseBlock-Stable and P1 as I believe this is breaking Chrome for many customers.

Bluecoat version is 6.5 for affected customer.

Jay Lee: Another customer using iBoss filtering solution is also seeing this issue.

This is going to be very problematic for many of our customers. We need to quickly work to provide a workaround solution for customers. Can we find a way to disable Chrome usage of TLS 1.3 at scale for customers (setting flag on each browser does not scale nor does it solve Chrome OS login screen issue where flags can't be set).

Some customers have noticed Chrome 56 does not always use TLS 1.3. For example, if they delete the "Local State" file in the user profile and restart Chrome, it will default to TLS 1.2 at least for awhile. Also, when Chrome OS devices are wiped and re-enrolled, they seem to default to 1.2 for awhile but then start using 1.3.

Can we get an explanation about how/when Chrome decides to use 1.3 and how customers can prevent it's use?

An affected IT person chimes in:

> Hello all.
>
> My environment is as follows:
> Chromebooks: Upwards of 50,000 (out of 120,000) Chromebooks have updated to OS56. Anywhere upwards of 30% of those 50,000 Chromebooks are stuck in a state of flickering between a login screen and a "Network not available" screen. Occasionally, you can see a SSL_HANDSHAKE_ERROR briefly at the login screen before switching back to the "Network not available" screen.
>
> PCs: 45,000 - 46,000 PCs have updated themselves to Chrome 56. Not all PCs are broken, but some are.
>
> BlueCoat 6.5 (which doesn't appear to have native support for TLS 1.3)

Someone else notes:

> Disconnect TLS 1.3 from base::FeatureList in M56.
>
> There are enterprises with broken firewalls that break when both sides negotiate TLS 1.3.

Jay returns with a summary (and [my] note, that this is a Catch-22 issue!)

For anyone following this issue, we are working on a Chrome update that should resolve by disabling TLS 1.3 in Chrome 56. In the meantime, there are a few other workarounds you may wish to try.

To be clear, ultimately this is an issue with proxies/firewalls that are not compatible with TLS 1.3. Please continue to work with your proxy/firewall vendor to update to a version that is compatible with TLS 1.3. A future version of Chrome will re-enable TLS 1.3.

Short-term workarounds:

- 1) On your internal DNS server, create a temporary A record that points clients4.google.com at 64.233.186.102. Once that's in place, restart Chrome / reboot Chrome devices a few times. It may take up to 30 minutes and a few restarts but devices should get the update to stop using TLS 1.2. **Important** be sure to remove the DNS A record once this is fixed. Leaving the record in place WILL BREAK DOWN THE LINE.

- 2) Have the user visit chrome://flags/#ssl-version-max and set to TLS 1.2. This works for Chrome users but not if the problem is occurring on Chrome OS login screen. **Important** be sure users turn this setting back to Default after leaving it on for 1-2 hours. Otherwise the user will not be able to use the more secure TLS 1.3 in the future and is left with a less secure profile.

- 3) Allow Chrome to connect directly to the Internet for connections to clients4.google.com. This could be done by connecting the device to a tethered phone, using a home network connection, disabling the firewall/proxy that is breaking TLS 1.3 or routing connections to clients4.google.com around this firewall/proxy. Once Chrome is able to connect to clients4.google.com, it should receive the update to disable TLS 1.3 automatically in 1-2 hours time. Restarts may be required.

CATCH-22: The point was... once you have received a Chrome 56 which starts using TLS v1.3, it can no longer connect to Google to receive a workable Chrome that DOESN'T support TLS v1.3!

Fingerprints to unlock iPhone? Judge says no.
https://www.scmagazine.com/fingerprints-to-unlock-iphone-judge-says-no/article/639939/
by Greg Masters, Managing Editor

A federal judge in Chicago issued an opinion on February 16 that would deny the government's attempt to force Apple device owners from providing a fingerprint to unlock their device.

The 14-page opinion and order, part of a child porn case, adds to the growing debate around individual rights to privacy and the needs of law enforcement to get past encryption technologies to further their investigations.

It boils down to Fourth Amendment protections against unreasonable search and seizure as well as the Fifth Amendment right to avoid self-incrimination.

In recent similar "forced fingerprinting" cases, prosecutors have argued that providing a fingerprint does not threaten an individual's Fifth Amendment right to not implicate oneself. A fingerprint provided as a means to identify an individual has, of course, been allowed in court. However, a fingerprint pressed into service as a means to unlock a user's smartphone is unfolding in courtrooms as an entirely different matter.

While Judge M. David Weisman stated in the court document, still sealed, that law enforcement did have probable cause to search a particular home, he drew the line when it came to "compelling individuals to provide their fingerprints to unlock an Apple electronic device."

The case in Chicago also riles privacy advocates who argue that requiring those swept up in an investigation to provide fingerprints to unlock a device raises Fourth Amendment issues against unreasonable searches and seizures.

Speaking with Ars Technica about the case, Abraham Rein, a Philadelphia-based tech lawyer, pointed out that "there is a big difference between using a fingerprint to identify a person and using one to gain access to a potentially vast trove of data about them and possibly about innocent third parties, too."

Others contend that the law's increasing attempts to force the issue likely will prompt technology companies to toughen up their devices, adding biometric verifications along with a passcode, for instance.

In any case, the extent of how far government officials can reach into individuals' possessions is still unfolding. And it's still an open question how policies will shape up under President Trump. While border agents have always had the right to search luggage and personal possessions, with the growth of smartphones and other personal digital devices, the officials' rights to search those possessions are currently in a constitutional grey area.

Some international travelers have been required to hand over their devices to U.S. Customs and Border Protection agents once they arrive into U.S. airports. While such incidents did increase fivefold in the last year of the Obama administration, privacy watchdogs have said they've seen a spike in complaints about searches of digital devices by border agents since Trump took office. Customs officials explain that travelers are carrying more devices than in the past, so it's only a perception that there's been an increase of searches.

My Take: Judges are coming down all over the place due to a current lack of clear statutory law. Even appeals courts are splitting on this. We're going to need ultimate clarification from the Supreme Court. And these issues are so important to the way our future unfolds that we need to hope that any decision is carefully considered and widely debated. I'd expect to see a huge number of amicus briefs presented by both Government and industry. We just need the right case to be argued.

Amazon refusing to hand over data on whether Alexa overheard a murder
https://arstechnica.com/tech-policy/2017/02/amazon-wont-disclose-if-alexa-witnessed-a-murder/
Meanwhile...
Remember that suspicious and suspected murder case in Bentonville, Arkansas late last November, where someone was found dead in a hot tub? Local police investigators have asked for all available evidence from the connected home's owners' IoT devices. They already obtained a dump from the IoT water meter which indicated that a HUGE and suspicious amount of water had been used in the wee hours of the night. But they want to know whether the owner's Amazon Echo device may have overhead anything illuminating.

ArsTechnica reports that: Amazon is balking at a search warrant seeking cloud-stored data from its Alexa Voice Service. Arkansas authorities want to examine the recorded voice and transcription data as part of a murder investigation. Among other things, the Seattle company claims that the recorded data from an Amazon Echo near a murder scene is protected by the First Amendment, as are the responses from the voice assistant itself.

Amazon said that the Bentonville Police Department is essentially going on a fishing expedition with a warrant that could chill speech and even the market for Echo devices and competing products. In a motion to quash the subpoena, the company said that because of the constitutional concerns at issue, the authorities need to demonstrate a "compelling need" for the information and must exhaust other avenues to acquire that data.

This case has generated a HUGE amount of publicity, putting heightened pressure on everyone on both sides of the struggle. Amazon needs to assure all of its customers that these devices are not surreptitious spy-bots and the detectives need to recover and analyze all possibly relevant evidence.

A Proof of Concept generates unnecessary concern

- Another demonstration of air-gapped computer data exfiltration.
- Press:
    - Watch Malware Steal Data From Air Gapped PC With Blinking Lights and a Drone
        - https://www.wired.com/2017/02/malware-sends-stolen-data-drone-just-pcs-blinking-led/

    - A computer's LED light can smuggle out data from the hard drive
        - http://www.upi.com/Science_News/2017/02/22/A-computers-LED-light-can-smuggle-out-data-from-the-hard-drive/4711487797215/

- The researchers found that the hard drive indicator LED of a PC can be controlled at up to 6,000 blinks per second, making it possible to transmit data to a remotely positioned optical receiver.

- In their proof-of-concept tests, the security researchers successfully transcribed and transmitted a computer's hard drive data using the flash of the LED light and Morse code-like sequences. The method moved data at speeds upwards of 4,000 bits per second.

- The said: "The LED is always blinking as it is searching and indexing, so no one suspects anything, even in the night. It's possible for the attacker to cause such fast blinking that a human never sees it."

Seclists.org: CVE-2016-9892 - Remote Code Execution as Root via ESET Endpoint Antivirus 6 for MacOS

- http://seclists.org/fulldisclosure/2017/Feb/68

- The ESET marketing material proudly boasts:
  "ESET Endpoint Antivirus for OS X delivers award-winning cross-platform protection for multi-platform environments. It protects against malware and spyware and shields end users from fake website phishing for sensitive information such as usernames, passwords or credit card details. Unauthorized devices can be blocked from the system entirely. The solution's highly intuitive interface allows for quick navigation."

- Unfortunately, however... Vulnerable versions of ESET Endpoint Antivirus 6 are statically linked with an outdated XML parsing library and do not perform proper server authentication, allowing for remote unauthenticated attackers to perform arbitrary code execution as root on vulnerable clients.

- The esets_daemon service, which runs as root, is statically linked with an outdated version of the POCO XML parser library (https://pocoproject.org/) -- version 1.4.6p1 from 2013-03-06. This version of POCO is based on Expat (http://expat.sourceforge.net/) version 2.0.1 from 2007-06-05, which has a publicly known XML parsing vulnerability (CVE-2016-0718) that allows for arbitrary code execution via malformed XML content.

  When ESET Endpoint Antivirus tries to activate its license, esets_daemon sends a request to https://edf.eset.com/edf. The esets_daemon service does not validate the web server's certificate, so a man-in-the-middle can intercept the request and respond using a self-signed HTTPS certificate. The esets_daemon service parses the response as an XML document, thereby allowing the attacker to supply malformed content and exploit CVE-2016-0718 to achieve arbitrary code execution as root.

Miscellany

AspiringLockpicker (@AspiringLockpic) / 2/27/17, 4:02 PM

- @SGgrc Is it ok to install an offline-only antivirus like clamwin?
  Or is that still issue-prone regarding OS hooks/delays etc?

Crypto education resources

- Crypto 101
  http://www.crypto101.io/
  https://9d0df72831e4b345bb93-4b37fd03e6af34f2323bb971f72f0c0d.ssl.cf5.rackcdn.com/Crypto101.pdf

- TerraSano (@TerraSano) 2/23/17, 7:37 PM
  - @SGgrc Thanks for the pointer to the cryptography book by Boneh. FWIW, he has a free (&excellent) crypto course thru coursera.com

- Five pages on Encryption
  - https://www.cs.princeton.edu/~felten/encryption_primer.pdf

- GRC's LinkFarm page now pulls all of these resources together.
  - (And Googling "LinkFarm" now returns GRC's page as the 2nd hit after Wikipedia's definition of the term. :)

Ryan McGinnis (@bigstormpicture) 2/23/17, 5:19 PM

- @SGgrc Good God #theexpanse is killing it! When did SyFy go from B-movie schlock to HBO quality space drama? They've gone plaid!

The Quiet Canine (aka The Protable Sound Blaster aka The Portable Dog Killer)

- https://www.grc.com/tqc/TheQuietCanine.htm

SpinRite

Glasair pilot (Glasair pilot) / 2/27/17, 1:29 PM (via Twitter DM)

- FYI, it's been almost two months since I did SR L2 on the two spinning drives (out of four) in my RAID 10. Zero Warnings since then, & array has not gone Critical since.

Sue's recent RAID-1 (Mirroring) experience

- Yay!!

The First SHA-1 Collision

Google Online Security Blog: Announcing the first SHA1 collision
https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

Bruce Schneier: SHA-1 Collision Found
https://www.schneier.com/blog/archives/2017/02/sha-1_collision.html

Press:

- Google Achieves First-Ever Successful SHA-1 Collision Attack
  http://thehackernews.com/2017/02/sha1-collision-attack.html

- I thought I'd write an update on git and SHA1, since the SHA1 collision attack
  https://plus.google.com/+LinusTorvalds/posts/7tp2gYWQugL

- Google Just Broke SHA-1 Encryption — One Of The Most Popular Cryptographic Hash Functions: https://fossbytes.com/google-broke-sha-1-encryption-hash/

- Watershed SHA1 collision just broke the WebKit repository, others may follow
  https://arstechnica.co.uk/security/2017/02/watershed-sha1-collision-just-broke-the-webkit-repository-others-may-follow/

- 'Re: SHA1 collisions found' - MARC
  http://marc.info/?l=git&m=148787047422954

VERY Cool chart/diagram here: Lifetimes of cryptographic hash functions

- http://valerieaurora.org/hash.html

Properties of a Cryptographic Hash / Wikipedia:

- It is deterministic so the same message always results in the same hash
- It is quick to compute the hash value for any given message
- It is infeasible to generate a message from its hash value except by trying all possible messages a small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value
- It is infeasible to find two different messages with the same hash value

As Google phrases it:
As a cryptographic requirement for wide-spread use, finding two messages that lead to the same digest should be computationally infeasible. Over time however, this requirement can fail due to attacks on the mathematical underpinnings of hash functions or to increases in computational power.

Today, more than 20 years after of SHA-1 was first introduced, we are announcing the first practical technique for generating a collision. This represents the culmination of two years of research that sprung from a collaboration between the CWI Institute in Amsterdam and Google. We've summarized how we went about generating a collision below. As a proof of the attack, we are releasing two PDFs that have identical SHA-1 hashes but different content.

For the tech community, our findings emphasize the necessity of sunsetting SHA-1 usage. Google has advocated the deprecation of SHA-1 for many years, particularly when it comes to signing TLS certificates. As early as 2014, the Chrome team announced that they would gradually phase out using SHA-1. We hope our practical attack on SHA-1 will cement that the protocol should no longer be considered secure. We hope that our practical attack against SHA-1 will finally convince the industry that it is urgent to move to safer alternatives such as SHA-256.

... Google: In 2013, Marc Stevens published a paper that outlined a theoretical approach to create a SHA-1 collision. We started by creating a PDF prefix specifically crafted to allow us to generate two documents with arbitrary distinct visual contents, but that would hash to the same SHA-1 digest. In building this theoretical attack in practice we had to overcome some new challenges. We then leveraged Google's technical expertise and cloud infrastructure to compute the collision which is one of the largest computations ever completed. Here are some numbers that give a sense of how large scale this computation was:

- Nine quintillion (9,223,372,036,854,775,808) SHA1 computations in total
- 6,500 years of CPU computation to complete the attack first phase
- 110 years of GPU computation to complete the second phase

While those numbers seem very large, the SHA-1 shattered attack is still more than 100,000 times faster than a brute force attack which remains impractical.

Detecting the use of this attack IS possible. *(!!!)*

No one would start using SHA-1 today... but as we know, legacy is hard to kill.

GIT is presently using SHA-1 but could easily switch to using 160 bits from a stronger hash.

https://shattered.io/
We have broken SHA-1 in practice.

This industry cryptographic hash function standard is used for digital signatures and file integrity verification, and protects a wide spectrum of digital assets, including credit card transactions, electronic documents, open-source software repositories and software updates.

It is now practically possible to craft two colliding PDF files and obtain a SHA-1 digital signature on the first PDF file which can also be abused as a valid signature on the second PDF file.

For example, by crafting the two colliding PDF files as two rental agreements with different rent, it is possible to trick someone to create a valid signature for a high-rent contract by having him or her sign a low-rent contract.