Security Now! #690 - 11-20-18

“Are Passwords Immortal?”

This week on Security Now!

This week we cover the action during last week's Pwn2Own Mobile hacking contest, as this year draws to a close to delve into the final last word on processor mis-design, we offer a very workable solution for unsupported Intel firmware upgrades for hostile environments, we look at a forthcoming Firefox breach alert feature, we cover the expected takeover of exposed Docker-offering servers, we note the recently announced successor to recently ratified HTTP/2, we cover a piece of 1.1.1.1 errata, close the loop with some of our podcast listeners, then we finish by considering the future of passwords using a thoughtful article written by Troy Hunt, a well-known Internet security figure and the creator of the popular "Have I Been Pwned" web service, among others.

Security News

Last week's Mobile Pwn2Own
... Was held last week the, November 13 & 14 during the PacSec 2018 Conference in Tokyo, Japan.

https://www.zerodayinitiative.com/blog/2018/11/13/pwn2own-tokyo-2018-day-one-results
https://www.zerodayinitiative.com/blog/2018/11/14/pwn2own-tokyo-2018-day-two-results-and-master-of-pwn

https://www.zerodayinitiative.com/Pwn2OwnTokyo2018Rules.html

Trend Micro is offering cash and prizes during the competition for vulnerabilities and exploitation techniques against the listed devices in the below categories. The first contestant to successfully compromise a device within the selected category will win the prizes for the category. All prizes are in US currency.

<<QUOTE>> The first day of Pwn2Own Tokyo 2018 has come to a close. Today saw some great action and amazing research as we awarded $225,000 USD and 48 Master of Pwn points. We had six successful exploits and purchased 13 bugs in total.

Our day began with Fluoroacetate (Amat Cama and Richard Zhu) successfully exploiting the Xiaomi Mi6 handset via NFC. Using the touch-to-connect feature, they forced the phone to open the web browser and navigate to their specially crafted webpage. During the demonstration, we didn't even realize that action was occurring until it was too late. In other words, a user would have no chance to prevent this action from happening in the real world. The webpage exploited an Out-Of-Bounds write in WebAssembly to get code execution on the phone. This earned them $30,000 USD and 6 Master of Pwn points.

The Fluoroacetate duo returned, this time targeting the Samsung Galaxy S9. They made quick work of it by using a heap overflow in the baseband component to get code execution. Baseband attacks are especially concerning, since someone can choose not join a Wi-Fi network, but they have no such control when connecting to baseband. The work earned them another $50,000 USD and 15 more points towards Master of Pwn.

Next up, Amat and Richard returned to the Short Distance category. This time, they were targeting the iPhone X over Wi-Fi. They used a pair of bugs – a JIT vulnerability in the web browser followed by an Out-Of-Bounds write for the sandbox escape and escalation. The successful demonstration earned them $60,000 USD more and 10 additional Master of Pwn points. This ends their first day of competition with $140,000 USD and a commanding lead for the Master of Pwn with 31 points.

Following that attempt, the team from MWR Labs combined three different bugs to successfully exploit the Samsung Galaxy S9 over Wi-Fi. They forced the phone to a captive portal without user interaction, then used an unsafe redirect and an unsafe application load to install their custom application. Although their first attempt failed, they nailed it on their second try to earn $30,000 USD and 6 more Master of Pwn points.

In our last entry of the day, Michael Contreras made sure his first Pwn2Own attempt was memorable. He wasted no time in exploiting a type confusion in JavaScript. In doing so, he earned himself $25,000 USD and 6 Master of Pwn points. We look forward to seeing him in future events. Excelsior!

##########

Our second day began with the Fluoroacetate duo of Amat Cama and Richard Zhu targeting the iPhone X in the browser category. After a stellar first day, they kicked off Day Two in style by combining a JIT bug in the browser along with an Out-Of-Bounds Access to exfiltrate data from the phone. For their demonstration, the data they chose happened to be a deleted picture, which certainly was a surprise to the person in the picture. The research earned them $50,000 and 8 more points towards Master of Pwn.

Next up, the MWR Labs team of Georgi Geshev, Fabi Beterke, and Rob Miller also targeted the iPhone X in the browser category. Unfortunately, they couldn't get their exploit chain to work within the time allotted, resulting in a failed attempt. However, they did have some great research, and we acquired the bugs through our normal ZDI program.

Following that, the Fluoroacetate team returned, this time targeting the web browser on the Xiaomi Mi6. They continued their successful contest by using an integer overflow in the JavaScript engine to exfiltrate a picture from the phone. They earned themselves another $25,000 USD and 6 Master of Pwn points.

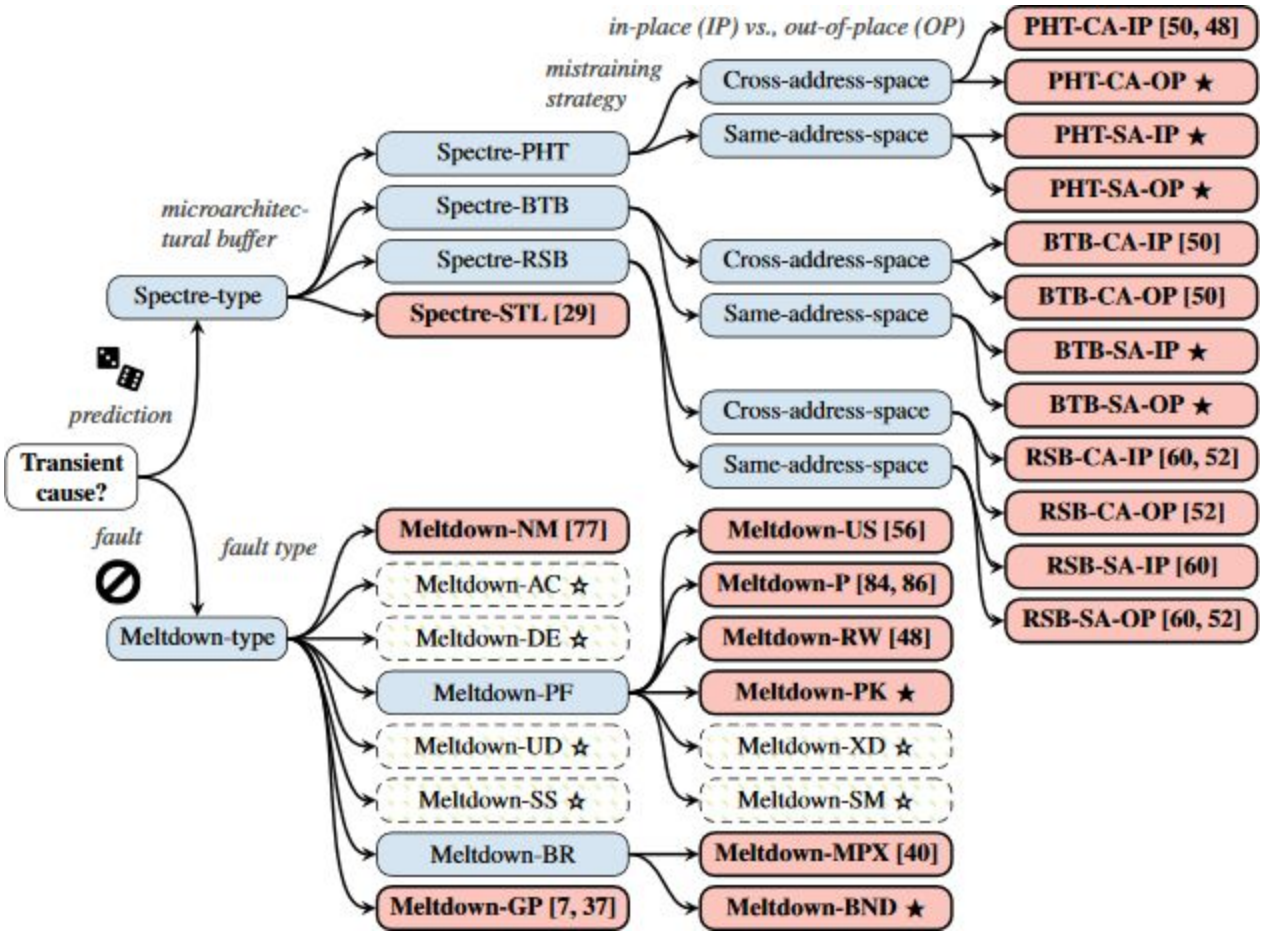
The Fluoroacetate team couldn't keep their momentum going throughout the entire competition, as their last entry fizzled out. They attempted to exploit the baseband on the iPhone X, but could not get their exploit working in the time allotted. Still, five out of six successful demonstrations is pretty remarkable. We're glad to see these two team up and hope to see more from them in the future.

Our final entry for this year's event saw the MWR Labs team target the web browser on the Xiaomi Mi6 handset. The team combined a download bug along with a silent app installation to load their custom application and exfiltrate pictures. This earned them another $25,000 USD and 6 additional points toward Master of Pwn.

That closes out Pwn2Own Tokyo for 2018. With 45 points and $215,000 USD total, we're happy to announce the Fluoroacetate duo of Amat Cama and Richard Zhu have earned the title Master of Pwn!

Overall, we awarded $325,000 USD total over the two day contest purchasing 18 0-day exploits. Onsite vendors have received the details of these bugs and now have 90 days to produce security patches to address the bugs we reported. Once these are made public, stay tuned to this blog for more details about some of the best and most interesting bugs we saw this week.

A Systematic Evaluation of Transient Execution Attacks and Defenses
https://arxiv.org/abs/1811.05441
https://arxiv.org/pdf/1811.05441.pdf



Universal transient execution attack classification tree with demonstrated attacks (red, bold), negative results (green, dashed), some first explored in this work (starred).

Abstract
Modern processor optimizations such as branch prediction and out-of-order execution are crucial for performance. Recent research on transient execution attacks including Spectre and Meltdown showed, however, that exception or branch misprediction events may leave secret-dependent traces in the CPU's microarchitectural state. This observation led to a proliferation of new Spectre and Meltdown attack variants and even more ad-hoc defenses (e.g., microcode and software patches).

Unfortunately, both the industry and academia are now focusing on finding efficient defenses that mostly address only one specific variant or exploitation methodology. This is highly problematic as the state-of-the-art provides only limited insight on residual attack surface and the completeness of the proposed defenses.

In this paper, we present a sound and extensible systematization of transient execution attacks. Our systematization uncovers 7 (new) transient execution attacks that have been overlooked and not been investigated so far. This includes 2 new Meltdown variants: Meltdown-PK on Intel, and Meltdown-BR on Intel and AMD. It also includes 5 new Spectre mistraining strategies. We

evaluate all 7 attacks in proof-of-concept implementations on 3 major processor vendors (Intel, AMD, ARM). Our systematization does not only yield a complete picture of the attack surface, but also allows a systematic evaluation of defenses. Through this systematic evaluation, we discover that we can still mount transient execution attacks that are supposed to be mitigated by rolled out patches.

1 Introduction

Processor performance over the last decades was continuously improved by shrinking processing technology and increasing clock frequencies, but physical limitations are already hindering this approach. To still increase the performance, vendors shifted the focus to increasing the number of cores and optimizing the instruction pipeline. Modern processors have deep pipelines allowing operations to be performed in parallel in different pipeline stages or different units of the execution stage. Many processors additionally have a mechanism which allows executing instructions not only in parallel but even out-of-order. These processors have a reordering element, which keeps track of all instructions and commits them in order, i.e., there is no functional difference to regular in-order execution. Intuitively, instructions executed on in-order and out-of-order pipelines cannot be executed, if they have a dependency on a previous instruction which has not been executed (and committed) yet. Hence, to keep the pipeline full at all times, it is essential to predict the control flow, data dependencies, and possibly even the actual data. Flushed instructions, those whose results are not made visible to the architectural level due to a roll-back, are called transient instructions [56, 50, 84]. On modern processors, virtually any instruction can raise a fault (e.g., page fault or general protection fault), requiring a roll-back. Already without prediction mechanisms, processors sometimes have to flush the pipeline, e.g., upon interrupts. With prediction mechanisms, there are more situations when partial pipeline flushes are necessary, namely on every misprediction. The pipeline flush reverts any architectural effects of instructions, ensuring functional correctness. Hence, the instructions are executed transiently (first they are, and then they vanish), i.e., we call this transient execution.

While the architectural effects and results of transient instructions are discarded, microarchitectural side effects remain beyond the transient execution. This is the foundation of Spectre [50], Meltdown [56], and Foreshadow [84]. These attacks exploit transient execution and encode secrets in the microarchitectural side effects (e.g., cache state) to transmit them (to the architectural level) to an attacker. The field of transient execution attacks emerged suddenly and grew rapidly, leading to a situation where people are not aware of all variants and their implications. This is apparent from the confusing naming scheme that already led to an arguably wrong classification of at least one attack [48]. Even more important, this confusion leads to misconceptions and wrong assumptions for defenses. Many defenses, e.g., focus exclusively on hindering exploitation of a specific covert channel, instead of addressing the microarchitectural root cause of the leakage [47, 45, 88, 50]. Other defenses critically rely on state-of-the-art CPU features that have not yet been thoroughly evaluated from a transient security perspective [83]. We also debunk implicit assumptions including that AMD processors are immune to Meltdown-type effects, or that serializing instructions mitigate Spectre Variant 1 on any CPU.

In this paper, we present a sound and extensible systematization of transient execution attacks, i.e., Spectre, Meltdown, Foreshadow, and related attacks. Using our universal decision tree, all known transient execution attacks were accurately classified through a universal and

unambiguous naming scheme (cf. Figure 1). The hierarchical and extensible nature of our classification methodology allows to easily identify residual attack surface, leading to 7 new transient execution attacks (Spectre and Meltdown variants) that we describe in this work. These 7 new attacks have been overlooked and not been investigated so far. Two of the attacks are Meltdown-BR, exploiting a Meltdown-type effect on the x86 bound instruction on Intel and AMD, and Meltdown-PK, exploiting a Meltdown-type effect on memory protection keys on Intel. The other 5 attacks are previously overlooked mistraining strategies for Spectre-PHT and Spectre-BTB attacks. We demonstrate all 7 attacks in practical proof-of-concept attacks on vulnerable code patterns and evaluate them on processors of Intel, ARM, and AMD.

We also provide a systematization of the state-of-the-art defenses. Based on this, we systematically evaluate defenses with practical experiments and theoretical arguments to show which work and which do not or cannot work. This systematic evaluation revealed that we can still mount transient execution attacks that are supposed to be mitigated by rolled out patches. Finally, we discuss how defenses can be designed to mitigate entire types of transient execution attacks.

The Intel Microcode Boot Loader (for entheusiasts)
https://www.ngohq.com/intel-microcode-boot-loader.html
https://www.techpowerup.com/forums/threads/intel-microcode-boot-loader.248858/

A developer has created a "pre-boot" patch for Intel microcode providing the latest Intel processor microcode for 392 Intel CPUs produced from 1996 to 2018.

This "Intel Microcode Boot Loader" provides a workaround for the "my BIOS hasn't been updated for Spectre and Meltdown and probably never will be" problem. It dynamically updates the microcode every time the system is powered up.booted. The Intel Microcode Boot Loader is based on Intel BIOS Implementation Test Suite (BITS) so that users no longer need to modify BIOS/UEFI ROMs to stay protected from security vulnerabilities, bugs and erratas.

Okay... Wait... What? What is "BITS" ??

BIOS Implementation Test Suite (BITS) for Processors
https://www.intel.com/content/www/us/en/support/articles/000007636/processors.html

What is the BIOS implementation test suite (BITS)?
The Intel BIOS Implementation Test Suite (BITS) provides a bootable pre-OS environment for testing BIOS and in particular the initialization of Intel processors, hardware, and technologies. BITS can verify your BIOS against many Intel recommendations. In addition, BITS includes Intel's official reference code as provided to BIOS, which you can use to override your BIOS's hardware initialization with a known-good configuration, and then boot an OS.

Who should use BITS?

You might want to use BITS if:

- You're a system or BIOS developer, and you want to validate that your system meets Intel's recommendations.
- You're an OS or application developer building on technologies provided by Intel platforms, and you want to check if your system (or one of your user's systems) has configured those technologies correctly.
- You're an advanced user or developer, and you want to check your BIOS to see if it configures Intel hardware correctly, and if not, to make a stronger case to your BIOS vendor to get it fixed.
- You need to poke hardware in a low-level way, and you need a pre-OS environment to work in to avoid OS interference.

BITS consists of a modified GRUB2 bootloader, with many additional commands to probe and manipulate hardware configuration, as well as scripts using these commands to test and reconfigure hardware.

BITS supports scripting via Python, and includes Python APIs to access various low-level functionality of the hardware platform, including ACPI, CPU and chipset registers, PCI, and PCI Express. You can write scripts to explore and test platform functionality, using the full power of Python in 32-bit ring 0, without an OS in the way, and without recompiling BITS or writing custom C code. See our Python scripting guide for more information.

https://biosbits.org/

Okay... So back to the "Intel Microcode Boot Loader" which was written on top of BITS:

This solution requires a permanently plugged-in USB flash drive with at least 25MB (or similar device) and BIOS/UEFI supporting boot from USB devices. Alternatively, advanced users can install it to a local drive on top of the System Reserved partition (see localdrive.txt for instructions).

Instructions:

1. Format a USB flash drive with FAT32 filesystem.
2. Extract the archive to the USB flash drive and run install.exe to make it bootable.
3. Enter the BIOS/UEFI, assign the USB flash drive as the 1st boot device and enable legacy boot mode.
4. The boot loader will regularly update the microcode and load the OS.

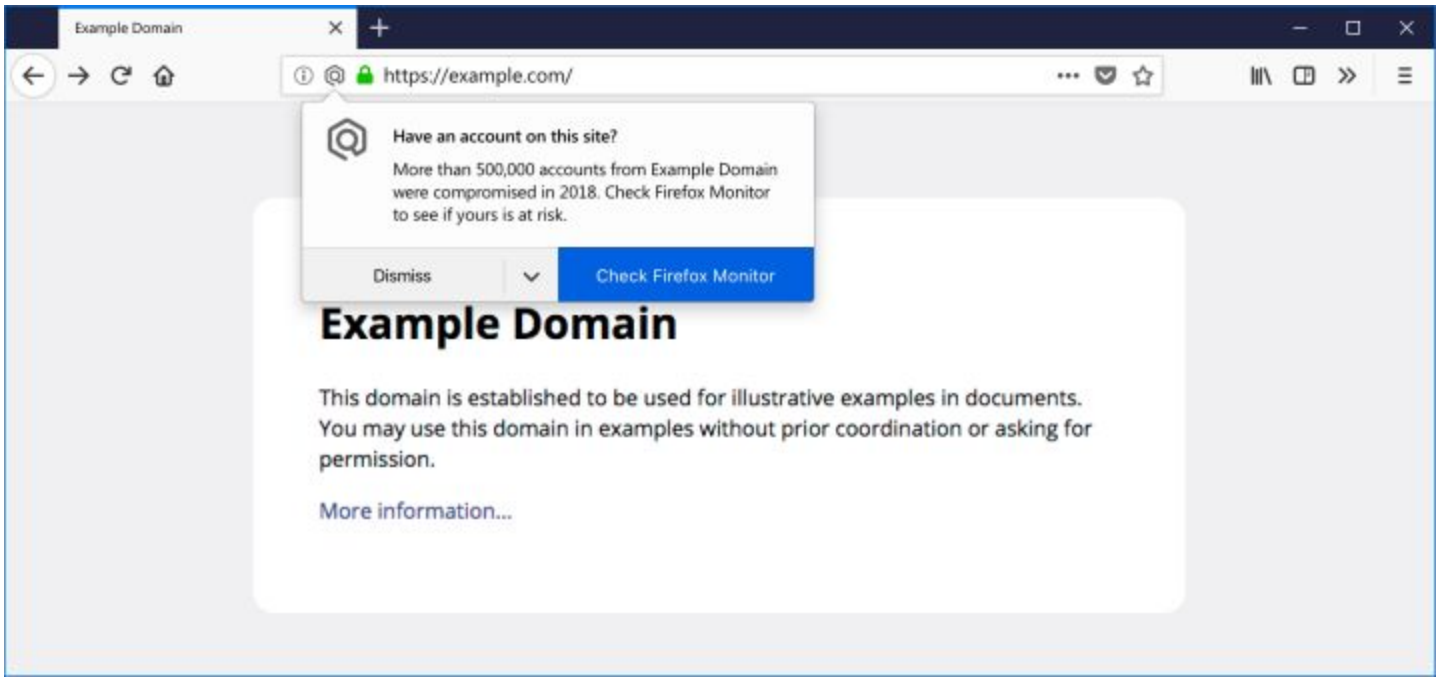
Notes:

- The microcode library is stored in the \boot\mcudb folder if you wish to update in the future.
- If you get 'Ucode not found' warning during installation, or plan to deploy on another PC, look for the correct ucode (by CPUID) in \boot\mcudb and copy it to \boot\mcu.

Changes (v0.5.1):

- Fixed a bug in the installer.
- Improved support for a local drive installation.
- Updated microcode database.

Firefox Monitor Launches in 26 Languages and Adds New Desktop Browser Feature
https://blog.mozilla.org/blog/2018/11/14/firefox-monitor-launches-in-26-languages-and-adds-new-desktop-browser-feature/



Last Wednesday Mozilla announced the addition of a new feature for FireFox: The first time (and only the first) time a FireFox user visits a site that has suffered from a publicly reported data breach within the past 12 months, a pop-up notification will appear notifying the user of a prior breach at that site.

The user can take note of and dismiss the notification, or can elect to immediately jump over to the FireFox Monitor site where they can confidentially enter their eMail address and have Troy Hunt's excellent "Have I been Pwned" site-as-a-service check for any exposure of the user's eMail. And, anyone not wishing to receive these alerts on any site, you can simply choose to "never show Firefox Monitor alerts" by clicking the dropdown arrow on the notification.

Mozilla states that this functionality will be gradually rolled out to FireFox user over the coming weeks.

We have not talked about FireFox monitor before. Aside from offering its own front end to Troy's Have I been pwned facility (which offers a web API to facilitate such access) users can sign up with the FireFox Monitor using their eMail address to receive proactive notification if a breach occurs involving their eMail address.

https://monitor.firefox.com/
https://haveibeenpwned.com/

As predicted: Unpatched Docker flaws open the door to Cryptominers
https://forums.juniper.net/t5/Threat-Research/Container-Malware-Miners-Go-Docker-Hunting-In-The-Cloud/ba-p/400587
https://securityboulevard.com/2018/11/juniper-networks-cryptomining-exploit-targeting-docker-containers/

We initially spoke of this problem and vulnerability a few weeks ago and now Juniper Threat Labs has recently discovered and reported on malware in the wild that searches for misconfigured, publicly exposed Docker services on the Internet and infects them with containers that run Monero miners.

Docker provides so-called "REST APIs" for management of its service, including the ability to create and start/stop containers. By default, Docker =ONLY= enables access to this API over Unix sockets, meaning within the same machine and NOT over the Internet. But, to enable remote access to the Docker service's REST APIs, << because, why not? >> Docker must be configured to also listen on TCP ports. The conventional ports used by Docker are 2375 and 2376 which, when enabled, provide unencrypted and unauthenticated access to the docker REST APIs.

Once a new host is infected it starts looking for other accessible hosts both on the public Internet and on any internal networks the host has access to... Which will typically be the internal corporate Intranet with the infected victim machine service as un unwitting bridge.

The entire infection chain is largely script based and Juniper describes it as "living off the land" by leveraging well-known, system-provided utilities to spread the infection and carry out its activities. Some of the system utilities that it uses are Docker, Wget, cURL, Bash, iproute2, MASSCAN, apt-get, yum, up2date, pacman, dpkg-query, systemd, and so forth.

Juniper's has dissected and described the operation of the malware in detail. It runs through several stages to get itself setup. In the third stage of the "auto.sh" shell script, it downloads MoneroOcean's Monero miner bash script hosted on Pastebin and executes it. Juniper notes that the downloaded script is pretty much the stock version straight from MoneroOcean's Github account. It download's XMRig's miner from the Github account and runs it as a systemd service.

Once that's done, the fourth stage of the auto.sh script, uses masscan to port-scan all network subnets connected to the infected host. It scans for hosts running docker daemon with open TCP ports 2375 and 2376, and dumps to a local file, local.txt, the list of new host IP addresses to infect.

Juniper concluded their write up of their findings by saying:

Cloud and containers are growing exponentially, but the exposure of services to the public internet has shown us that tons of services are misconfigured or loosely configured, leaving them vulnerable to be exploited by malicious parties.

In the same realm, misconfigured Docker services on the cloud are inviting cryptominers to utilize the vast computing resources of the cloud. But, it doesn't have to stop there for these malicious actors. Apart from the availability of the massive computing power, the short-lived

nature of containers means malicious parties can use containers to create temporary and effective attack points in their infection chain. Imagine a botnet that brings up containers running their bot on-demand, runs a DDoS and then shuts down the container, thereby reducing the footprint of its presence.

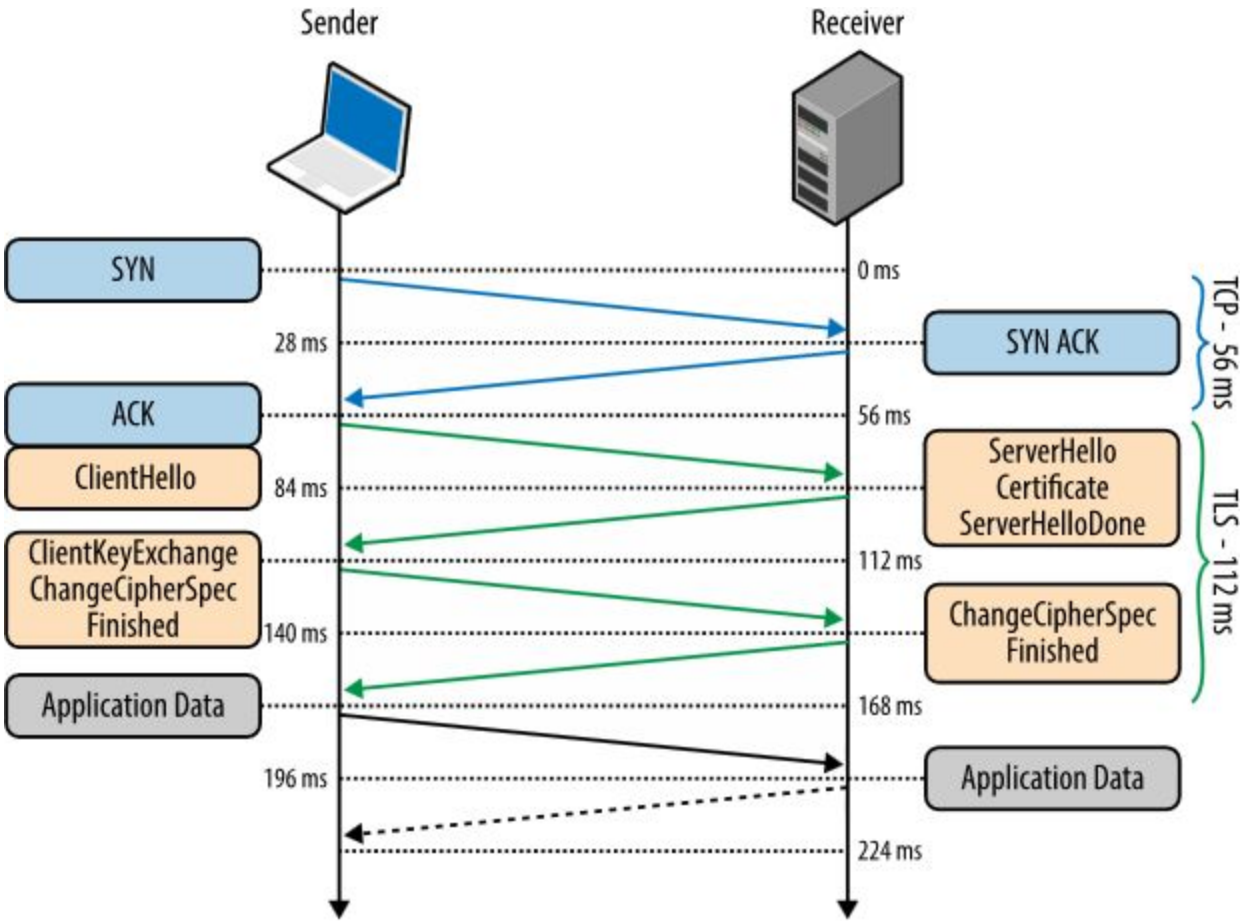
Correctly configuring network-facing services, instilling authenticated access to exposed ports and opening TCP ports only if necessary can alleviate some of these concerns.

HTTP/3 will be UDP

Mark Nottingham, who is the chair of both the HTTP working group and the QUIC working group for the IETF proposed renaming HTTP-over-QUIC to HTTP/3, and the proposal appears to have been broadly accepted. HTTP/3 will have QUIC as an essential, integral feature such that HTTP/3 will always use QUIC as its network protocol.

So let's back up a bit here.

As we know, traditional secure connections over the Internet begin with a DNS lookup to map the domain name to the Internet address -- originally IPv4 and increasingly IPv6. But address lookup is typically cached and it's not really about connections. So it's out-of-scope for this.



But, once we have an IP address, multiple packets are sent back and forth to first establish a TCP connection. Sequence numbers are exchanged and IP-level features are negotiated between the endpoints.

At that point we have a connection. So the client (typically a user's web browser) sends a Client Hello packet to the server to announce their supported cipher suites and to notify the server of the hosting machine name whose certificate they need.

The server rummages around to find the proper certificate and sends its Server Hello packet back to the client containing its own Hello and the Certificate. Both of these handshakes also contain cryptographic nonces to enable the establishment of a single common secure secret.

The client finishes the client key exchange, and confirms the server's choice of cipher by returning a ChangeCipherSpec Finished message. And the server confirms the receipt of the client's ChangeCipherSpec Finished by returning its own.

And, finally, at that point, a secure, authenticated and privately encrypted tunnel has been established allowing the client to send its request for a web page to the server.

Believe it or not, QUIC, which stands for ""Quick UDP Internet Connections" is able, under the proper conditions and preconditions, to achieve the same... with a single packet from client to server.

And for this we have Google to thank. Recall that Google's earlier SPDY technology also proved itself worthy and formed the basis of HTTP/2.

HTTP-over-QUIC is a rewrite of the HTTP protocol that uses Google's QUIC instead of TCP as its base. QUIC is Google's complete rewrite and rethink of that entire protocol stack to combine everything -- TCP, UDP, TLS and HTTP/2 into a single amalgam.

Google has proposed that QUIC might eventually replace both TCP and UDP as the new protocol of choice for moving binary data across the Internet. Tests have demonstrated that QUIC is both faster and more secure because it is an encrypted-by-default system with the newly ratified TLS v1.3 protocol built in.

QUIC was proposed as a draft standard at the IETF in 2015, and HTTP-over-QUIC, a re-write of HTTP on top of QUIC instead of TCP, was proposed a year later, in July 2016.

HTTP-over-QUIC support was added to Chrome 29 and Opera 16 and is also supported by the LiteSpeed web servers and Facebook has also started adopting the technology.

Errata

How does https://1.1.1.1 give us a valid cert?

SpinRite

Christian Alexandrov / @Diabolikkant / 2:32am · 19 Nov 2018 · Twitter Web Client

@SGgrc one short spinrite testimonial, if a HDD wants to die, it will die. My regular use of spinrite on a drive gave me a lot of warning, so I had the time to save all my data which I care about. One more lesson for people: regular use of spinrite warns you when something is going to happen.

Closing The Loop

Terry Danial / @terryontech

@SGgrc since #SQRL tokens are linked to web site nam is there a solution in #SQRL for the problem that occurs when a web property changes its name (e.g. United to Continental or Ofoto.com to KodakGallery.com)?

John W Baxter / @johnbaxter

Hi, Steve. Do you have any information on how long external SSD devices can sustain their contents while not plugged in to power. Specifically the Samsung T5. Clearly they aren't permanent, as the capacitors have to be leaking charge at some rate. Two weeks seems to be OK. I haven't found information at Samsung or in reviews. (At two weeks I could be running through spares way too fast, but I think I would have seen complaints.) One solution might be to store the devices plugged in on power, but at some point the better choice would seem to be spinning drives.

James @Jr_McG1

Listening to this week’s episode of security now. Could you produce an ‘app specific public key’ that could be given to the bots that need use of your SQRL identity, and have the main protocol authorise the use of that key for a specific app when you are connected to an authenticated session. You could then manage on a per site basis the apps that have authority to use your identity on that site, much the same way we use app specific passwords on google and other services now. You can then ‘prune’ your app specific keys in the sqrl app with any revocations being communicated to the site via the protocol and signed with your master identity to authorise the revocation.

yodar44 @yodar44

Steve, in sn689 you focused on bitLocker on SSD's. Do the same concerns apply to bitLocker on spinning drives?

CPH / @CoreyHartless

Steve, Windows 7 BitLocker is unaffected by the hardware encryption problem because Windows 7 BitLocker doesn’t support leveraging hardware encryption in the first place, so VeraCrypt is not the only recourse there.

Greg W Hampton @fwdesperado
Steve, I'm listening to this week's SN episode and your segment on Mozilla's laboratory plug-in was perfectly timed. I just received a security report from our corporate security team on one of the servers that I manage. One of the findings was that 4 of the sites hosted on the server didn't have a security policy listed. I was nervous about remediating the findings as theses are internet facing production sites. I now have a way of creating an appropriate policy which won't break the sites. Thanks again to you and Leo for everything that you do.

Andreas Göb @a_goeb

Hi Steve!

Long time listener (yada, yada)…

Just heard you talk about Content Security Policy (CSP). You can actually put your policy into the site but set it to "report only" mode:
https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP#Testing_your_policy

You give it a reporting URL and all your users' browsers tell you what *would* break if the policy were to be enforced, so you know what to fix before you even activate it. Scott Helme and Troy Hunt run a service at report-uri.com that can receive these reports and provides analytics - free for up to 10,000 reports / month.

Currently, Google is building upon that idea to let Chrome report all kinds of stuff to a dedicated endpoint, e.g. if your site uses deprecated features, or has network errors or crashes the browser:
https://developers.google.com/web/updates/2018/09/reportingapi#reportypes

Mikael Falkvidd @mfalkvidd
re SSD encryption: the key *can not* be derived from the password. The password *must* be used to unlock the key. Otherwise, it would be impossible to ever change the password. And it would be impossible to start using a password without wiping the entire drive. I might have misunderstood your description (/rant) - if that's the case I'm sorry but I would love a clarification.

Tinzien @Tinzien
Hi Steve! In the past you advocated using Windows Defender plus MalwareBytes. With new Win Def sandbox, are you suggesting no MalwareBytes is needed/is a vector for problems or is this an omission? Thank you for your continued great work!

Troy Hunt ... on Passwords…

Three days ago, Saturday afternoon, I put what I've been working on for the past three months online to be pounded on and commented upon by the wonderful group of testers and techies in GRC's SQRL newsgroup. So far I've fixed a few bugs and I have a short ToDo list of features to add or tweak. This was an important piece of work since it defined a minimal and workable generic SQRL Service Provider API which allows a webserver to query an external provider for all of the site's SQRL support. The provider still needs to be present on the webserver's domain, but this allows for a clean externalization and boundary between SQRL's authentication functions and an existing website.

Fifteen days ago, on November 5th, Troy Hunt -- the "Have I Been Pwned?" guy, posted an article which he titled:

"Here's Why [Insert Thing Here] Is Not a Password Killer"

https://www.troyhunt.com/heres-why-insert-thing-here-is-not-a-password-killer/

These days, I get a lot of messages from people on security related things. Often it's related to data breaches or sloppy behaviour on behalf of some online service playing fast and loose with HTTPS or passwords or some other easily observable security posture. But on a fairly regular basis, I get an email from someone which effectively boils down to this:

> Hey, have you seen [insert thing here]? It's totally going to kill passwords!

No, it's not... and to save myself from repeating the same message over and over again, I want to articulate precisely why passwords have a lot of life left in them yet. But firstly, let me provide a high-level overview of the sort of product I'm talking about and I'll begin with recognising the problem it's trying to solve:

People suck at passwords. I know, massive shock right? They suck at making genuinely strong ones, they suck at making unique ones and they suck at handling them in a secure fashion. This leads to everything from simple account takeover (someone else now controls their eBay or their Spotify or whatever else), to financial damages (goods or services bought or sold under their identity) to full on data breaches (many of these occur due to admins reusing credentials). There is no escaping the fact that passwords remain high-risk security propositions for the vast majority of people. Part of the solution to this is to give people the controls to do password-based authentication better, for example by using a password manager and enabling 2FA. But others believe that passwords themselves have to go completely to solve the problem which brings us to proposed alternatives.

I don't want to single any one product out here because the piece I'm writing is bigger than that, so let's talk about patterns instead. I'm referring to passwordless solutions that involves things like QR codes, pictorial representations, 3rd party mobile apps, dedicated hardware devices or "magic" links sent via email. I'm sure there are others but for the purpose of this post, any pattern that doesn't involve entering a username and a password into a couple of input fields is

in scope. To their credit, some of these solutions are genuinely very good - technically very good - but what proponents of them seem to regularly miss is that "technically" isn't enough. Despite their respective merits, every one of these solutions has a massive shortcoming that severely limits their viability and it's something they simply can't compete with:

Despite its many flaws, the one thing that the humble password has going for it over technically superior alternatives is that everyone understands how to use it. Everyone.

This is where we need to recognise that decisions around things like auth schemes go well beyond technology merits alone. Arguably, the same could be said about any security control and I've made the point many times before that these things need to be looked at from a very balanced viewpoint. There are merits and there are deficiencies and unless you can recognise both (regardless of how much you agree with them), it's going to be hard to arrive at the best outcome.

Let me put this in perspective: assume you're tasked with building a new system which has a requirement for registration and subsequently, authentication. You go to the Marketing Manager and say "hey, there's this great product called [insert thing here] that replaces passwords and all you have to do to sign in is...". And you've already lost the argument because the foremost thing on the Marketing Manager's mind is reducing friction. Their number one priority is to get people signing up to the service and using it because ultimately, that's what generates revenue or increases brand awareness or customer loyalty or achieves whatever the objective was for creating the service in the first place. As soon as you ask people to start doing something they're not familiar with, the risk of them simply not going through with it amplifies and defeats the whole point of having the service in the first place.

This isn't a new discussion either, the one about considering usability alongside security. In fact, we have this discussion every time we build a system that defines minimum criteria for passwords; yes, a min of 20 chars would make for much stronger passwords but no, we very rarely do this because we actually want customers! If you look at the minimum password length requirements by large online services you can see the results of this discussion covering a fairly broad spread with Google and Microsoft demanding at least 8 characters, Amazon and Twitter wanting 6 and Netflix requiring... 4. But I'm hesitant to berate Netflix for what seems like an extremely low number because they're also dealing with the usability challenge that is people logging on to TVs with remote controls. The point of all this is that usability is an absolutely essential attribute of the auth discussion.

What I often find when I have these discussions is a myopic focus on technical merits. I'll give you an example from earlier last year where someone reached out and espoused the virtues of the solution they'd built. They were emphatic that passwords were no longer required due to the merits of [insert thing here] and were frustrated that the companies they were approaching weren't entertaining the idea of using their product. I replied and explained pretty much what's outlined above - the conversation is going to get shut down as soon as you start asking companies to impose friction on their users but try as I might, they simply couldn't get the message. "What barrier? There is no barrier!" They went on to say that companies not willing to embrace products like this and educate their users about alternative auth schemes are the real problem and that they should adjust their behaviour accordingly. I countered with what remains a point that's very hard to argue against:

> If your product is so awesome, have you stopped to consider why no one is using it?

Now in fairness, it may not be precisely "no one" but in this case and so many of the other [insert things here], I'd never seen them in use before and I do tend to get around the internet a bit. Maybe they're used in very niche corners of the web, the point is that none of these products are exactly taking the industry by storm and there's a very simple reason for that: there's a fundamental usability problem. This particular discussion ended when they replied with this:

> I think it is only negativity that doesn't allow positiveness to excel

Ugh. I'm negative about stuff that's no good, yes. I dropped out of the discussion at that point.

Almost a year ago, I travelled to Washington DC and sat in front of a room full of congressmen and congresswomen and explained why knowledge-based authentication (KBA) was such a problem in the age of the data breach. I was asked to testify because of my experience in dealing with data breaches, many of which exposed personal data attributes such as people's date of birth. You know, the thing companies ask you for in order to verify that you are who you say you are! We all recognise the flaws in using static KBA (knowledge of something that can't be changed), but just in case the penny hasn't yet dropped, do a find for "dates of birth" on the list of pwned websites in Have I Been Pwned. So why do we still use such a clearly fallible means of identity verification? For precisely the same reason we still use the humble password and that's simply because every single person knows how to use it.

This is why passwords aren't going anywhere in the foreseeable future and why [insert thing here] isn't going to kill them. No amount of focusing on how bad passwords are or how many accounts have been breached or what it costs when people can't access their accounts is going to change that. Nor will the technical prowess of [insert thing here] change the discussion because it simply can't compete with passwords on that one metric organisations are so focused on: usability. Sure, there'll be edge cases and certainly there remain scenarios where higher-friction can be justified due to either the nature of the asset being protected or the demographic of the audience, but you're not about to see your everyday e-commerce, social media or even banking sites changing en mass.

Now, one more thing: if I don't mention biometrics and WebAuthn they'll continually show up in the comments anyway. On the biometrics front, I'm a big supporter of things like Face ID and Windows Hello (I love my Logitech BRIO for that). But they don't replace passwords, rather provide you with an alternate means of authenticating. I still have my Apple account password and my Microsoft password, I just don't use them as frequently. WebAuthn has the potential to be awesome, not least of which because it's a W3C initiative and not a vendor pushing their cyber thing. But it's also extremely early days and even then, as with [insert things here], it will lead to a change in process that brings with it friction. The difference though - the great hope - is that it might redefine authentication to online services in an open, standardised way and ultimately achieve broad adoption. But that's many years out yet.

So what are we left with? What's actually being recommended and, indeed, adopted? Well to start with, there's lots of great guidance from the likes of the National Cyber Security Centre (NCSC) in the UK and the National Institute of Standards and Technology (NIST) in the US. I summarised many of the highlights in my post last year about Passwords Evolved: Authentication Guidance for the Modern Era. As a result of that piece (and following one of the NIST recommendations), I subsequently launched the free Pwned Passwords service which is being used by a heap of online entities including EVE Online and GitHub. This (along with many of the other NCSC or NIST recommendations) improves the password situation rather than solves it and it does it without fundamentally changing the way people authenticate. Every single solution I've seen that claims to "solve the password problem" just adds another challenge in its place thus introducing a new set of problems.

This is why [insert thing here] is not a password killer and why, for the foreseeable future, we're just going to have to continue getting better at the one authentication scheme that everyone knows how to use: passwords.

###

Now, I'll admit that I got a kick out of the fact that the very first reply posted to Troy's article was by a David A. Leedom • 15 days ago

"Has anyone looked at SQRL?"

And a large portion of the dialog was a back and forth with some online names I recognized from GRC's SQRL newsgroup and Twitter. The point is skepticism is a healthy trait. And it doesn't offend or annoy me in the slightest. But what I have also observed is that anyone who has actually seen SQRL and experienced it is immediately and irrevocably converted into a true believer. I think that what confuses people so much is how patient I am. I deeply believe in the truism that we only get one chance to make a first impression. So I'm working as hard as I can, day and night so that everyone's first impression will be all that it can and should be.

But a few points to Troy's article...
As we know: SQRL is not a product. It's a solution and a protocol which, by design, no one owns.

I have simply built an implementation of a SQRL client and now several implementations of SQRL servers. That was done to work out all of the details and edge-cases. If I were to do it again, I would have built my SQRL client as a web extension, since SQRL should ideally be integrated into every web browser. But web extensions were not unified back when I began this, and there IS a web extension in the works by someone else for Firefox and Chrome.

And as for no one owning it... There is also a working Android client over on Github whose SQRL implementation is currently available in Arabic, Chinese, Dutch, English, French, German, Hebrew, Japanese, Norwegian, Russian, Spanish and Swedish. There's a native client coming along for Linux, and there's a working client for iOS. Github also has a crypto library for SQRL and there are server implementations for Java and PHP in the works. All of these other people are not nuts. They're not crazy. They have one thing in common that, for the moment, separates them from the rest of the world: They have seen SQRL work first hand and they have all realized what it means.

And, yeah... I'm beginning to get excited because it won't be long now until everyone will have the chance to play with it and see -- and judge -- for themselves what this might mean.

And as for REPLACING what we have now? SQRL is quite happy to coexist alongside usernames and passwords, to simply be there as an option and benefit offered by websites that choose to offer it. At any such website, if someone is a SQRL user with a SQRL identity and wishes to "associate" their SQRL identity with their existing identity at a website, they can easily do so. And from then on they can just use SQRL there. And once they become comfortable with how SQRL works, and have their identity spread around a few machines and phones for safety to that it cannot be lost (and even better, after they have backed up their identity), they can set an option in their client to ask the websites they use SQRL with to ONLY accept their identity authentication via SQRL.

Mostly, though, once you experience the use of SQRL for yourself firsthand -- an experience you will all have soon since it is really not far off -- I believe you'll see that what it offers is as near to zero friction logon as is possible to achieve. Yes, there's a bit of setup required. But it's one time. And there's no subscription, no expiration, no third party involvement, and no one is making any money from this. Everyone who is involved with SQRL believes that people will rationally, out of their own self interest -- because usernames and passwords have become such a problem -- over time choose to use it and migrate to it. And once you have you really cannot go back.

I agree with Troy 100%, that nothing yet has been good enough to supplant usernames and passwords, and that they are here because they always have been. And I also agree that any real meaningful change takes time. We know about inertia. That's the lesson we learn here about everything. How long as IPv6 been around? And TLS v1.0 which we are just now starting to deprecate is 20 years old.

But in concluding, my main point is that until and unless there IS a perfect alternative available -- just available -- (where "perfect" in this context means open, free, zero friction and answers every problem) -- until such a thing exists nothing WILL ever change. The change cannot be forced. We know that. I agree with Troy 100%. And I will be happy to let SQRL speak for itself. I think we're going to find it's able to.

~30~