Security Now! #757 - 03-10-20

The Fuzzy Bench

This week on Security Now!

This week we consider the new time-limited offers being made for free telecommuting tools, the continuing success of the DoD's "please come hack us" program, another take on the dilemma and reality of Android device security, some unwelcome news about AMD processor side-channel vulnerabilities, a new potentially serious and uncorrectable flaw in Intel processors, a 9.8-rated critical vulnerability in Linux system networking, a stand back and watch the fireworks forced termination of TLS v1.0 and v1.1, and the evolution of the SETI @ Home project after 19 years of distributed radio signal number crunching. We then touch on a bit of miscellany and finish by looking at a new and open initiative launched by Google to uniformly benchmark the performance of security Fuzzers.

:)

Security News

Microsoft, Google, LogMen & Cisco offer limited-time free use of Telecommuting Tools
The cynic in me thinks that perhaps this is marketing opportunism: Using a global pandemic to get people hooked on the benefits of stay-at-home working through a time-limited offer. But on the other hand, we all know how much resistance there is to change. And in a competitive market, if any one of them made their services available for free the others would be missing out on a true opportunity to expand their user base -- permanently.

So, yeah... with employees either being quarantined at home after international travel, or being encouraged to work from home while the COVID-19 Coronavirus situation stabilizes, Microsoft, Google, LogMeIn, and Cisco are all offering free use of their meeting, collaboration, and remote work tools. And, since avoiding a daily commute can be a **very** positive thing for oneself, the environment and everyone else, I would expect that a percentage of those who first make the jump because they really have no choice at the moment might choose to embrace that option in the longer term future.

Microsoft Teams: No charge for six months
Microsoft EVP and President of Microsoft Global Sales, Marketing & Operations, tweeted:

Google offers free access to Hangouts Meet for G Suite users
Not to be left behind, Google also announced this week that they are offering G Suite and G Suite for Education customers free access to their Hangouts Meet video-conferencing features.

This includes:

- Larger meetings, for up to 250 participants per call
- Live streaming for up to 100,000 viewers within a domain
- The ability to record meetings and save them to Google Drive

Google's announcement stated that: "These features are typically available in the Enterprise edition of G Suite and in G Suite Enterprise for Education, and will be available at no additional cost to all customers until July 1, 2020."

LogMeIn offers free Emergency Remote Work Kits
https://blog.gotomeeting.com/coronavirus-disruptions-and-support/

Bill Wagnar, LogMeIn's CEO posted a blog titled: "Coronavirus Disruptions: An Offer of Support to our Communities and Customers" stating… "Starting immediately, we will be offering our critical front-line service providers with free, organization-wide use of many LogMeIn products for 3 months through the availability of Emergency Remote Work Kits. These kits will include solutions for meetings and video conferencing, webinars and virtual events, IT support and management of remote employee devices and apps, as well as remote access to devices in multiple locations. For example, the "Meet" Emergency Remote Work Kit will provide eligible organizations with a free site-wide license of GoToMeeting for 3 months."

Cisco offers free enhanced Webex licenses
And, Cisco has modified its free Webex license to now support meetings with no time limit and up to 100 participants. And, in addition, Cisco is also offering free 90-day licenses to businesses that are not currently Webex customers. Their announcement stated: "Additionally, through our partners and the Cisco sales team, we are providing free 90-day licenses to businesses who are not Webex customers in this time of need. We're also helping existing customers meet their rapidly changing needs as they enable a much larger number of remote workers by expanding their usage at no additional cost." And now that Cisco has finally patched the high-severity vulnerability in Webex which allowed strangers to barge into password-protected meetings without any authentication, Cisco's offer might also be worth considering.

So, Cisco is 90 days, LogMeIn is 3 months, Google is until July 1st. So assuming a start time of April 1st, that's also 3 months. Only Microsoft is offering double those free-use duration. And it would be annoying to get all setup on Google, LogMeIn or Cisco only to find that Corona is taking longer than expected to calm down. Our illustrious president Trump has famously explained that this whole thing will be disappearing like magic once the weather warms up. Which appears to be how Google, LogMeIn and Cisco set their free-use schedules. But if it should turn out that the virus missed Trump's announcement, it might make more sense to run with Microsoft's 6-month plan if their "Teams" offering meets your company's needs.

The DoD's "please come hack us" program is panning out bigtime!

/// UNCLASSIFIED ///
DoD CYBER CRIME CENTER (DC3) Vulnerability Disclosure Program (VDP) ANNUAL REPORT 2019
Volume 1

https://www.dc3.mil/data/uploads/vdp/annual-reports/2019_vdp_annualmetricvol1.pdf

Our listeners may recall that four years ago the US Department of Defense first invited white hat hackers to hack into its systems in what was dubbed "Hack the Pentagon." Today, after four years of success with the initially controversial program, the Pentagon continues to ask hackers to hit them with everything they've got. And, really... What could possibly be more fun than attacking the online presence of the US Defense Department -- with their official permission so that you won't be convicted?!?

The just-released report states that after 4 years these industrious white hats are submitting more vulnerability reports than ever.

The DoD's Department of Defense Cyber Crime Center (DC3) handles cybersecurity for the DoD, and is responsible for tasks including cyber technical training and vulnerability sharing. It also runs the DoD's Vulnerability Disclosure Program (VDP).

The VDP emerged from the Hack the Pentagon bug bounty program that the military ran in 2016. That initiative was so successful that it continues to invite hackers to play with its systems. Last year, as we reported at the time, the Air Force even brought an F-15 to Defcon for hackers to tinker with. The F-15 Eagle, for those not up on the US Military's air combat, is described as an all-weather, extremely maneuverable, tactical fighter designed to gain and maintain air superiority in aerial combat. So last summer the hackers had one of those to play with. Next year the DoD plans to bring a satellite!

So, with this newly published report we have some numbers. The report reveals that it processed 4,013 vulnerability reports from a total of 1,460 white hat researchers. It validated 2,836 (nearly 3/4 of them) for mitigation. This past year was the busiest year for bug reports, representing a 21.7% increase over 2017 and bringing the total number of bug reports to 12,489.

The report explained:

> *"These vulnerabilities were previously unknown to the DoD and not found by automated network scanning software, red teams, manual configuration checks, or cyber inspections. Without DoD VDP there is a good chance those vulnerabilities would persist to this date, or worse, be active conduits for exploitation by our adversaries."*

What more does anyone need to know? This is a crazy-good idea. Information exposure bugs were the most common type reported during the year, followed by violation of secure design principles, cross-site scripting flaws, business logic errors, and open redirects which can be used to launch a phishing attack.

Moving forward, the DoD Cyber Crime Center wants to expand the scope of the program beyond DoD websites to cover any DoD information system. It also wants to partner with the Defense Counterintelligence and Security Agency (DCSA) to create what it calls a defense industrial base (DIB) VDP program to secure the DoD's supply chain.

This is all such rare and right thinking on the part of our government. And this experience should go a long way toward further spreading the "bug bounty" concept throughout other areas of the Us's massive government bureaucracy.

The DoD's bug bounty program is being managed by HackerOne, the absolute best place to do these things. And during the program's first pilot test which ran for only four weeks, from April 18th to May 12th of 2016, the results were said to exceed all of their wildest expectations. The first report arrived 13 minutes after the program opened, 200 reports were received within 6 hours, a total of 1410 hackers participated, and $75,000 in verified bug bounties were paid.

At this point, I think there's ample proof to suggest that any sizable company ought to be participating in the encouragement of white hat hackers to discover and report important vulnerabilities. Everything we have seen tells us that vulnerabilities will remain undiscovered until someone is motivated to look. And a few thousand dollars is a small price to pay for the discovery and resolution of a remotely targetable vulnerability.

The Android Security Dilemma

Sophos Security's Naked Security site posted some useful detail and insight to one of the worries that I often voice here. So I wanted to share what they had to say:

Their piece published yesterday was titled: "One billion Android smartphones racking up security flaws"
https://nakedsecurity.sophos.com/2020/03/09/one-billion-android-smartphones-racking-up-security-flaws/

How long do Android smartphones and tablets continue to receive security updates after they're purchased? The slightly shocking answer is barely two years, and that's assuming you bought the handset when it was first released. Even Google's own Pixel devices max out at three years.

Many millions of users hang on to their Android devices for much longer, which raises questions about their ongoing security as the number of serious vulnerabilities continues to grow.

Add up all the Android handsets no longer being updated and you get big numbers – according to Google's developer dashboard last May, almost 40% of Android users still use handsets running versions 5.0 to version 7.0, which haven't been updated for between one and four years. One in ten run something even older than that, equivalent to one billion devices.

The point is brought home by new testing from consumer group Which?, discovering that it was possible to infect popular older handsets mainly running Android 7.0 – the Motorola X, Samsung Galaxy A5, Sony Xperia Z2, Google Nexus 5 (LG), and the Samsung Galaxy S6 – with mobile malware.

All of the above were vulnerable to a recently discovered Bluetooth flaw known as BlueFrag, and to the Joker strain of malware from 2017. The older the device, the more easily it could be infected – Sony's Xperia Z2, running Android 4.4.2, was vulnerable to the StageFright flaw from 2015.

I just checked Amazon. The Sony Xperia Z2 is for sale on Amazon -- new. It is, and will always be, vulnerable to the very worrisome Stagefright vulnerability where just the receipt of a specially crafted MMS message is sufficient to remotely take over the phone.

Google recently had to remove 1,700 apps containing Joker (aka Bread) from its Play Store, only the latest in an increasingly desperate rearguard action against malware being hosted under its nose. It's not simply that these devices aren't getting security fixes, but older models also miss out on a bundle of security and privacy enhancements that Google has added to versions 9 and 10.

Kate Bevan, a Computing editor for Which.co.uk (and formerly of Naked Security), said:

"It's very concerning that expensive Android devices have such a short shelf life before they lose security support – leaving millions of users at risk of serious consequences if they fall victim to hackers."

Bevan raised the interesting point that the idea that a device might only get updates for two years will come as news to most Android users. She said: "Google and phone manufacturers need to be upfront about security updates, with clear information about how long they will last and what customers should do when they run out."

Google has issued the same response to several media outlets in response to the report: "We're dedicated to improving security for Android devices every day. We provide security updates with bug fixes and other protections every month, and continually work with hardware and carrier partners to ensure that Android users have a fast, safe experience with their devices."

I'll note that one problem is awareness. Anyone still using Windows 7 will have received a big full-screen warning that it is no longer safe to use this operating system. But no smartphone suddenly warns its user that it is no longer safe to be used. They just quietly stop receiving security updates... if they ever received any at all after the original sale.

Sophos continues: In truth, users are being squeezed between two forces. On the one hand, Google is determined to drive the evolution of Android for competitive reasons, releasing a new version every year. On the other side are manufacturers, eager to keep people upgrading to new models on the pretext that the older ones won't run these updated versions of Android (which is not always true).

Security sits somewhere between the two, and despite attempted reforms by Google in recent years to make security fixes happen on a monthly cycle, the reality is some way from that ideal.

Eventually, there comes a time to discard an old device, but for most users that will be longer than two years. To drive home the point about flaws, the March 2020 Android security bulletin patched a MediaTek flaw, CVE-2020-0069, which has being actively exploited in the wild for several months. And yet MediaTek had a fix for the flaw last May, but device makers didn't apply it. Even now that it's namechecked in Google's update, it could take months to percolate through to devices because updates happen so slowly. And this is a flaw known to be exploited in the wild.

AMD Processors get some unwelcome but necessary side-channel attack scrutiny
Our prolific friends from the Graz University of Technology and Research Institute of Computer Science, and Random Systems (IRISA), responsibly disclosed the vulnerabilities to AMD back in August 2019.

This past Saturday, March 7th, AMD updated their "AMD Product Security" page with a response to the unwelcome news: https://www.amd.com/en/corporate/product-security

> We are aware of a new white paper that claims potential security exploits in AMD CPUs, whereby a malicious actor could manipulate a cache-related feature to potentially transmit user data in an unintended way. The researchers then pair this data path with known and mitigated software or speculative execution side channel vulnerabilities. AMD believes these are not new speculation-based attacks.

The researchers, who were some of the original discoverers of the Spectre, Meltdown and ZombieLoad vulnerabilities disagree with AMD's assessment. Their research paper is titled: "Take A Way: Exploring the Security Implications of AMD'sCache Way Predictors" https://mlq.me/download/takeaway.pdf

> ABSTRACT
> To optimize the energy consumption and performance of their CPUs, AMD introduced a way predictor for the L1-data (L1D) cache to predict in which cache way a certain address is located. Consequently, only this way is accessed, significantly reducing the power consumption of the processor.
>
> In this paper, we are the first to exploit the cache way predictor. We reverse-engineered AMD's L1D cache way predictor in microarchitectures from 2011 to 2019, resulting in two new attack techniques. With Collide+Probe, an attacker can monitor a victim's memory accesses without knowledge of physical addresses or shared memory when timesharing a logical core. With Load+Reload, we exploit the "way predictor" to obtain highly-accurate memory-access traces of victims on the same physical core. While Load+Reload relies on shared memory, it does not invalidate the cache line, allowing stealthier attacks that do not induce any last-level-cache evictions.
>
> We evaluate our new side channel in different attack scenarios. We demonstrate a covert channel with up to 588.9 kB/s, which we also use in a Spectre attack to exfiltrate secret data from the kernel. Furthermore, we present a key-recovery attack from a vulnerable cryptographic implementation. We also show an entropy-reducing attack on ASLR of the kernel of a fully patched Linux system, the hypervisor, and our own address space from JavaScript. Finally, we propose countermeasures in software and hardware mitigating the presented attacks.

So, while AMD skirted some of the last two years of speculative execution damage, that was largely because researchers were primarily targeting Intel's performance-enhancing micro-architecture. But AMD is by no means immune to similar attacks which leverage anything that a processor might do to speculatively increase its performance by storing the past.

Annnnnnnnd... Intel also has some serious new trouble on its hands.
On March 5th last week, Positive Technologies posted a report titled: "Intel x86 Root of Trust: loss of trust" and, bizarrely, this came as no surprise to Intel. They were apparently hoping that no one would notice. I wouldn't be surprised if lawsuits arise.

What has effectively happened is that a flaw which is very similar to Apple's unpatchable CheckM8 boot ROM flaw has been found to be present in all Intel processors produced in the past 5 years.

https://blog.ptsecurity.com/2020/03/intelx86-root-of-trust-loss-of-trust.html#more

Positive Technologies' disclosure begins as follows:
The scenario that Intel system architects, engineers, and security specialists perhaps feared most is now a reality. A vulnerability has been found in the ROM of the Intel Converged Security and Management Engine (CSME). This vulnerability jeopardizes everything Intel has done to build the root of trust and lay a solid security foundation on the company's platforms. The problem is not only that it is impossible to fix firmware errors that are hard-coded in the Mask ROM of microprocessors and chipsets. The larger worry is that, because this vulnerability allows a compromise at the hardware level, it destroys the chain of trust for the platform as a whole.

Positive Technologies specialists have discovered an error in Intel hardware, as well as an error in Intel CSME firmware at the very early stages of the subsystem's operation, in its boot ROM. Intel CSME is responsible for initial authentication of Intel-based systems by loading and verifying all other firmware for modern platforms. For instance, Intel CSME interacts with CPU microcode to authenticate UEFI BIOS firmware using BootGuard. Intel CSME also loads and verifies the firmware of the Power Management Controller responsible for supplying power to Intel chipset components.

Even more importantly, Intel CSME is the cryptographic basis for hardware security technologies developed by Intel and used everywhere, such as DRM, TPM, and Intel Identity Protection. In its firmware, Intel CSME implements EPID (Enhanced Privacy ID). EPID is a procedure for remote attestation of trusted systems that allows identifying individual computers unambiguously and anonymously, which has a number of uses: these include protecting digital content, securing financial transactions, and performing IoT attestation. Intel CSME firmware also implements the TPM software module, which allows storing encryption keys without needing an additional TPM chip—and many computers do not have such chips.

Intel tried to make this root of trust as secure as possible. Intel's security is designed so that even arbitrary code execution in any Intel CSME firmware module would not jeopardize the root cryptographic key (Chipset Key), but only the specific functions of that particular module.

Unfortunately, no security system is perfect. Like all security architectures, Intel's had a weakness: the boot ROM, in this case. An early-stage vulnerability in ROM enables control over reading of the Chipset Key and generation of all other encryption keys. One of these keys is for the Integrity Control Value Blob (ICVB). With this key, attackers can forge the code of any Intel CSME firmware module in a way that authenticity checks cannot detect. This is functionally equivalent to a breach of the private key for the Intel CSME firmware digital signature, but limited to a specific platform.

The EPID issue is not too bad for the time being because the Chipset Key is stored inside the platform in the One-Time Programmable (OTP) Memory, and is encrypted. To fully compromise EPID, hackers would need to extract the hardware key used to encrypt the Chipset Key, which resides in Secure Key Storage (SKS). However, this key is not platform-specific. A single key is used for an entire generation of Intel chipsets. And since the ROM vulnerability allows seizing control of code execution before the hardware key generation mechanism in the SKS is locked, and the ROM vulnerability cannot be fixed, we believe that extracting this key is only a matter of time. When this happens, utter chaos will reign. Hardware IDs will be forged, digital content will be extracted, and data from encrypted hard disks will be decrypted.

The vulnerability discovered by Positive Technologies affects the Intel CSME boot ROM on all Intel chipsets and SoCs available today other than Ice Point (Generation 10). The vulnerability allows extracting the Chipset Key and manipulating part of the hardware key and the process of its generation. However, currently it is not possible to obtain that key's hardware component (which is hard-coded in the SKS) directly. The vulnerability also sets the stage for arbitrary code execution with zero-level privileges in Intel CSME.

We will provide more technical details in a full-length white paper to be published soon. We should point out that when our specialists contacted Intel to report the vulnerability, Intel said the company was already aware of it (CVE-2019-0090). Intel understands they cannot fix the vulnerability in the ROM of existing hardware. So they are trying to block all possible exploitation vectors. The patch for CVE-2019-0090 addresses only one potential attack vector, involving the Integrated Sensors Hub (ISH). We think there might be many ways to exploit this vulnerability in ROM. Some of them might require local access; others need physical access.

= = = = = = =

So what does this mean?

It means that all Intel processors released in the past 5 years contain an unpatchable vulnerability that could allow hackers to compromise almost every hardware-enabled security technology that is designed to shield sensitive data of users even when a system is compromised.

As with Apple's CheckM8, because the vulnerability resides in ROM it cannot be repaired without replacing the chip.

Intel CSME is a separate security micro-controller incorporated into the processors that provides an isolated execution environment protected from the host operating system running on the main CPU.

The CSME is responsible for the initial authentication of Intel-based systems by loading and verifying firmware components, root of trust based secure boot, and also cryptographically authenticates the BIOS, Microsoft System Guard, BitLocker, and other security features.

Intel obfuscated and dramatically downplayed this problem when it was previously patched last year. At that time, Intel described it as just a privilege escalation and arbitrary code execution in Intel CSME firmware modules.

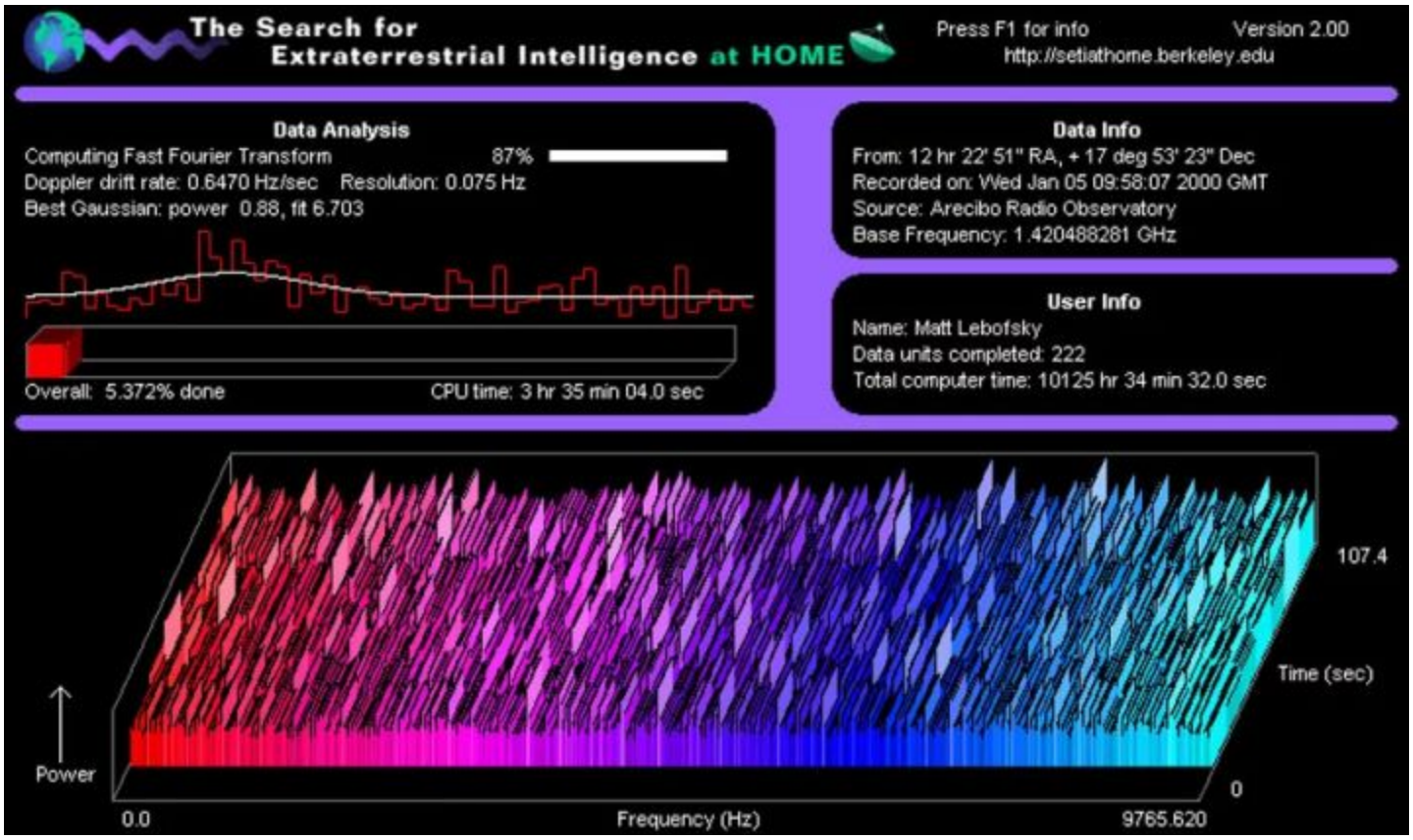
Remember, the researchers wrote: "... since the ROM vulnerability allows seizing control of code execution before the hardware key generation mechanism is locked, and the ROM vulnerability cannot be fixed, we believe that extracting this key is only a matter of time. When this happens, utter chaos will reign. Hardware IDs will be forged, digital content will be extracted, and data from encrypted hard disks will be decrypted.

Once the root of the chain of trust is compromised all of the security technologies which rely upon it are also compromised. So digital rights management (DRM), firmware-based Trusted Platform Module (TPM), and Identity Protection Technology (IPT) are now at risk.

The security patches released by Intel last year are incomplete. They do not and can not entirely prevent sophisticated attacks, leaving millions of systems at the risk of digital attacks that may be impossible to detect and patch. Since the ROM flaw can be exploited by an attacker with physical access before the system boots, NO possible downstream software update patch can prevent local attack.

Only Intel's very latest 10th-generation CPUs are unaffected (because THOSE they were able to fix on the production line.)

SETI@home shuts down its distributed computing project after 21 years.



SETI, of course, is the Search for ExtraTerrestrial Intelligence. (In light of recent events, the team has decided to turn their search inward and start searching for any signs of **terrestrial** intelligence. :)

Okay, well, no... But at the end of this month, exactly 3 weeks from now on March 31st, they will be evolving their very cool, 21-year long project. They're not giving up, it's time to figure out what, if anything, they have found during this period.

As many of us know, SETI@home is a distributed computing project where for the past 21 years, and thanks to the global Internet, volunteers spread across the world have been contributing their spare CPU resources to analyze raw radio data received from the Arecibo radio telescope in Puerto Rico and West Virginia's Green Bank Telescope for signs of coherent and organized radio emanations which, we presume and hope, would invariably arise from non-Earthly intelligeNT beings.

During this 21-year run, the UC Berkeley SETI Research Center, since May 17th, 1999 has been sending small chunks of raw data, each forming a "job" to volunteers.

The raw data are siphoned "piggyback" or "passively" while those two telescopes are used for other scientific programs. The data are digitized, stored, and sent to the SETI@home facility. The data are then parsed into small chunks divided in frequency and time, and analyzed to search for signals, where "signals" is defined as variations which cannot be ascribed to noise, and hence contain information.

Using distributed computing, SETI@home sends the millions of chunks of data to be analyzed off-site by home computers, to have those computers report the results. So what would be a massive problem in data analysis is reduced to a reasonable one thanks to the equivalent massive combined, donated and widely distributed computing power.

The software searches for five types of signals that distinguish them from noise:

- Spikes in power spectra
- Gaussian rises and falls in transmission power, possibly representing the telescope beam's main lobe passing over a radio source
- Triplets – three power spikes in a row
- Pulsing signals that possibly represent a narrowband digital-style transmission
- And autocorrelation detects signal waveforms.

So, here's what the folks at SETI wrote:

Titled: “SETI@home hibernation”
On March 31, the volunteer computing portion of SETI@home will stop distributing work and will go into hibernation.

We're doing this for two reasons:

1) Scientifically, we're at the point of diminishing returns; basically, we've processed all the data we need for now.

2) It's a lot of work for us to manage the distributed processing of data. We need to focus on completing the back-end analysis of the results we already have, and writing this up in a scientific journal paper.

We're extremely grateful to all of our volunteers for supporting us in many ways during the past 20 years. Without you there would be no SETI@home. We're excited to finish up our original science project, and we look forward to what comes next.

Critical PPP Daemon Flaw Opens Most Linux Systems to Remote Hackers

Last Thursday, US-CERT issued an advisory warning users of a newly discovered and dangerous 17-year-old remote code execution vulnerability affecting the PPP daemon (pppd) which is always present, if not usually running, on most Linux based OSes. And PPP may be powering the communications of Linux-based embedded networking devices.

So, PPP, or Point-To-Point protocol is seriously old school. It was originally used with dial-up modems (remember those?) and later with DSL and some VPN links. So, while it's not in super-active use, it is old and, as with most things on the Internet, never really dies until it is deliberately removed.

Consequently, this won't affect most Linux users, but it is yet another legacy protocol that has been found to be critically buggy, and which, if it WERE to be in use and not updated, could open any system -- at either the client or the server end of a PPP connection -- to hostile takeover.

The critical stack buffer overflow vulnerability was discovered by an IOActive security researcher to exist due to a logical error in the Extensible Authentication Protocol (EAP) packet parser of the pppd software. As its name suggests, it's an extension that provides support for additional authentication methods in PPP connections.

The vulnerability is being tracked as CVE-2020-8597 and it carries a CVSS Score of a whopping 9.8 because it can be exploited by unauthenticated attackers to remotely execute arbitrary code on affected systems and take full control over them. However, as I noted, you first need to be actually using PPP for there to be any danger. So long as PPP is never used, you're never in danger.

But, if PPP is in use, all an attacker needs to do is to send an unsolicited malformed EAP packet to a vulnerable ppp client or a server over a direct serial link, ISDN, Ethernet, SSH, SOcket CAT, PPTP, GPRS, or ATM networks.

And note that many things that maintain a persistent connection, like an ATM machine, which has also been around for years and is running an older version of Linux might well be vulnerable.

And since pppd often runs with high privileges and works in conjunction with kernel drivers, the flaw could allow attackers to potentially execute malicious code with the system or root-level privileges.

The Advisory says: "This vulnerability is due to an error in validating the size of the input before copying the supplied data into memory. As the validation of the data size is incorrect, arbitrary data can be copied into memory and cause memory corruption, possibly leading to the execution of unwanted code."
"The vulnerability is in the logic of the eap parsing code, specifically in the eap_request() and

eap_response() functions in eap.c that are called by a network input handler."

"It is incorrect to assume that pppd is not vulnerable if EAP is not enabled or EAP has not been negotiated by a remote peer using a secret or passphrase. This is due to the fact that an authenticated attacker may still be able to send unsolicited EAP packet to trigger the buffer overflow."

So... the researcher said that Point-to-Point Protocol Daemon versions 2.4.2 through 2.4.8 — all versions released in the last 17 years — are vulnerable to this longstanding remote code execution vulnerability.

Linux distros are updating their pppd daemons, so for those whose systems are being proactively maintained, this problem will disappear soon. But, again, we absolutely know that a great many machines which are serving turnkey embedded functions will not be updated and will now become subject to targeted attacks. And IF such a machine were on the network of any security-sensitive entity -- any major enterprise, power grid control, nuclear reactor, etc. -- this could open any of those otherwise secure systems to attack.

And as I mentioned above, the vulnerable protocol is also present in some turnkey product offerings. The research named Cisco's CallManager, TP-LINK products, the OpenWRT Embedded OS and Synology's products. So... another reason to keep on top of updates. All appearances suggest that exploiting this will not be difficult.

Speaking of deliberately terminating legacy protocols...
Back in the spring of 2018, the developers behind all of our major web browsers -- Safari, Chrome, Firefox and Edge -- gathered following the release of TLS v1.3 and jointly announced in October of that year their intention to remove support for TLS v1.0 and v1.1 early THIS year.

The first stage of this deliberate protocol deprecation began last year when browsers began labeling sites that were using TLS 1.0 and TLS 1.1 with a "Not Secure" indicator in the URL address bar and the lock icon, hinting to users that the HTTPS connection was not as secure as they might imagine. Since the maximum TLS version a site is offering was not something any end-user had control over, the hope was that users would complain to websites and that those sites would proactively move to support newer TLS versions, namely v1.2 and v1.3.

How'd that work out??
Well… today, more than 850,000 websites are still offering the use of protocols no higher than TLS 1.0 or 1.1 protocols. This includes sites of major banks, governments, news organizations, telecoms, e-commerce stores, and internet communities, according to a recent report published by the well known UK technology firm Netcraft. All the 850,000 websites offer HTTPS, but only over versions of TLS that are now regarded as weak and insecure.

Time flies, and TLS v1.0 and v1.1 have grown old and long in the tooth. 1.0 was released back in 1996 and v1.1 was 14 years ago in 2006. The protocols support the use of many weak cryptographic algorithms and are vulnerable to a series of cryptographic attacks that we have carefully described during this podcast's 12 year history, including BEAST, LUCKY 13, SWEET 32, CRIME, and POODLE. To varying degrees, these attacks allow attackers to decrypt portions of

HTTPS and access bits of a user's plaintext web traffic. None are critical vulnerabilities, but given that we now have far superior alternatives, it really is time to move on.

So, to that end, later this month, our web browsers will switch from showing a hidden warning which needs to be deliberately opened, to showing full-page errors when users access sites that use TLS 1.0 or TLS 1.1.

This should be interesting. <g>

Miscellany

SandSara:

> Hi Steve!
>
> Huge fan of the podcast! Kind of surreal to hear you talk about my project after listening to you talk so much over the years! I really appreciate all the nice things you said, if you ever get to building your own big table and have any questions, don't hesitate to reach out!
>
> I just upgraded your order to include both the RGB lighting and tempered glass upgrades.
>
> I also wanted to do a special offer for your listeners. Any order that comes through this link will get the RGB upgrade for free. (You can redirect https://grc.sc/sand to that one if you want)
>
> Thanks again!

https://grc.sc/sand
So far: 3445 listeners used that link, 17 of whom decided that, like you and me, Leo, they had to have one! There are some of each still remaining: 6 of the circular birch, 6 of the circular dark walnut, 9 of the star in birch and 22 of the star in dark walnut.

Freddfarkl @freddfarkl
Can you make mention of a specific Vitamin C liquid that seems to be so well liked?
https://grc.sc/liquid

The Fuzzy Bench

Posted last week to the Google Open Source Blog:
https://opensource.googleblog.com/2020/03/fuzzbench-fuzzer-benchmarking-as-service.html

"FuzzBench: Fuzzer Benchmarking as a Service"
Monday, March 2, 2020

First of all, let's review fuzzing so that we're all on the same page... WikiPedia defines fuzzing this way:

> "Fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program. The program is then monitored for exceptions such as crashes, failing built-in code assertions, or potential memory leaks. Typically, fuzzers are used to test programs that take structured inputs. This structure is specified, e.g., in a file format or protocol and distinguishes valid from invalid input. An effective fuzzer generates semi-valid inputs that are "valid enough" in that they are not directly rejected by the parser, but do create unexpected behaviors deeper in the program and are "invalid enough" to expose corner cases that have not been properly dealt with."

So, fuzzing is an interesting way to uncover bugs in programs. Rather than having a single highly-skilled hacker with knowledge of all past vulnerabilities carefully and methodically trying this or that in attempt to exploit something that might be possible, fuzzing is like the thousand monkeys all pounding on typewriters to see whether any of them might, by pure happenstance, hit upon something novel and useful.

In another of Google's "working to make the world a better place because we have plenty of money, so why not?", last week they posted the explanation of their latest initiative:
We are excited to launch FuzzBench, a fully automated, open source, free service for evaluating fuzzers. The goal of FuzzBench is to make it painless to rigorously evaluate fuzzing research and make fuzzing research easier for the community to adopt.

Fuzzing is an important bug finding technique. At Google, we've found tens of thousands of bugs with fuzzers like libFuzzer and AFL. There are numerous research papers that either improve upon these tools (e.g. MOpt-AFL, AFLFast, etc) or introduce new techniques (e.g. Driller, QSYM, etc) for bug finding. However, it is hard to know how well these new tools and techniques generalize on a large set of real world programs. Though research normally includes evaluations, these often have shortcomings—they don't use a large and diverse set of real world benchmarks, use few trials, use short trials, or lack statistical tests to illustrate if findings are significant. This is understandable since full scale experiments can be prohibitively expensive for researchers. For example, a 24-hour, 10-trial, 10 fuzzer, 20 benchmark experiment would require 2,000 CPUs to complete in a day.

To help solve these issues the OSS-Fuzz team is launching FuzzBench, a fully automated, open source, free service. FuzzBench provides a framework for painlessly evaluating fuzzers in a reproducible way. To use FuzzBench, researchers can simply integrate a fuzzer and FuzzBench will run an experiment for 24 hours with many trials and real world benchmarks. Based on data

from this experiment, FuzzBench will produce a report comparing the performance of the fuzzer to others and give insights into the strengths and weaknesses of each fuzzer. This should allow researchers to focus more of their time on perfecting techniques and less time setting up evaluations and dealing with existing fuzzers.

Integrating a fuzzer with FuzzBench is simple, as most integrations are less than 50 lines of code. Once a fuzzer is integrated, it can fuzz almost all 250+ OSS-Fuzz projects out of the box. We have already integrated ten fuzzers, including AFL, LibFuzzer, Honggfuzz, and several academic projects such as QSYM and Eclipser.

https://github.com/google/fuzzbench/tree/master/fuzzers

Reports include statistical tests to give an idea how likely it is that performance differences between fuzzers are simply due to chance, as well as the raw data so researchers can do their own analysis. Performance is determined by the amount of covered program edges, though we plan on adding crashes as a performance metric. You can view a sample report here:

https://www.fuzzbench.com/reports/sample/index.html
(Check out the sample... Fuzzers were run against all sorts of well-known code bases.)
curl, FreeType, jsoncpp, libjpeg, libpcap, libpng, libxml2, openssl, openthread, php, sqlite3, vorbis, woff2)

How to Participate

Our goal is to develop FuzzBench with community contributions and input so that it becomes the gold standard for fuzzer evaluation. We invite members of the fuzzing research community to contribute their fuzzers and techniques, even while they are in development. Better evaluations will lead to more adoption and greater impact for fuzzing research.

We also encourage contributions of better ideas and techniques for evaluating fuzzers. Though we have made some progress on this problem, we have not solved it and we need the community's help in developing these best practices.

https://github.com/google/FuzzBench

So, yeah... another big tip of the hat to Google for so willingly and usefully giving back to the community and, really, to the world. Though everyone knows that fuzzing is not the end-all solution for hardening our software and making it bulletproof, it inarguably provides another avenue into providing real world code security.

Efficient Fuzzing is not easy. It's the province of University academics and there's still a lot of progress to be made. So having an open platform for testing and comparing fuzzing innovations can only help academics to test and hone their new ideas.

Bravo, Google!