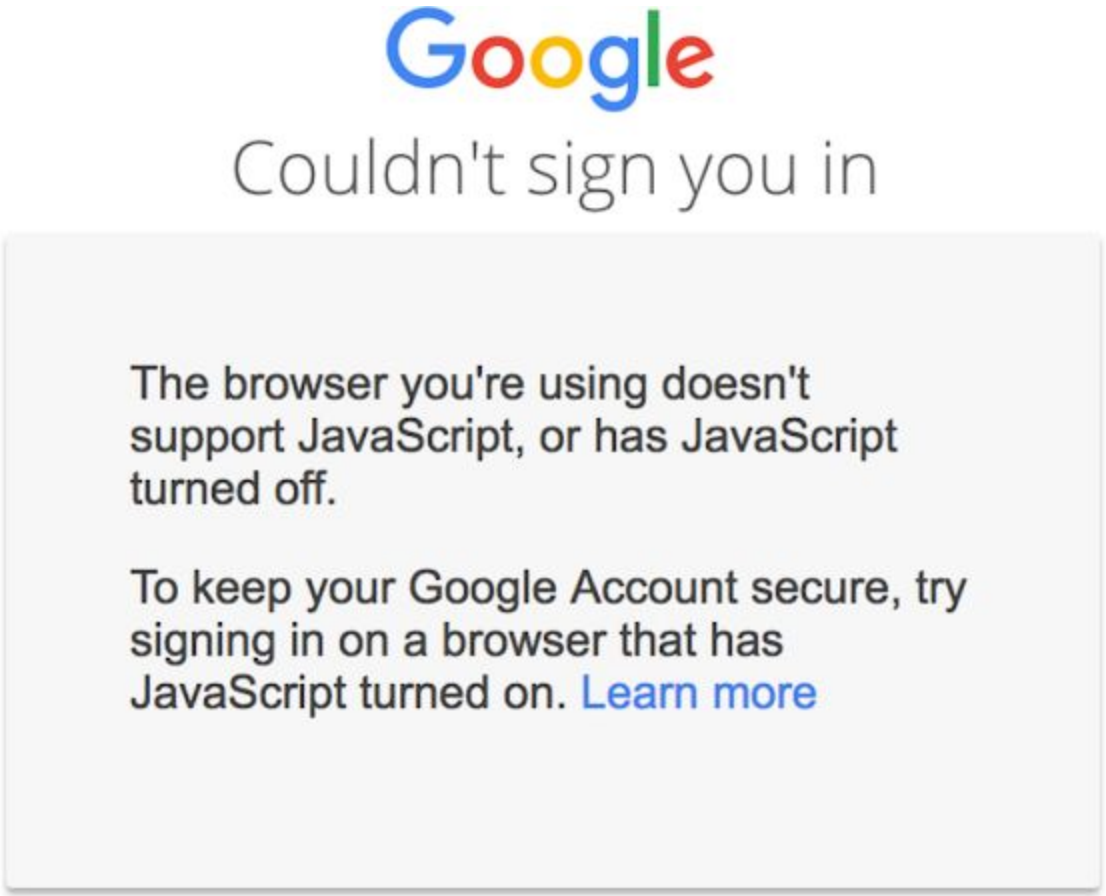
Security Now! #688 - 11-06-18

“PortSmash”

This week on Security Now!

This week we discuss the new “BleedingBit” Bluetooth flaws, JavaScript no longer being optional with Google, a new Microsoft Edge browser 0-day, Windows Defender plays in its own sandbox, Microsoft and SysInternals news, the further evolution of the CAPTCHA, the 30th anniversary of the Internet's first worm, a bizarre requirement of Ransomware, a nice new bit of security non-tech from Apple, some closing-the-loop feedback from our listeners… and then we take a close look at the impact and implication of the new “PortSmash” attack against Intel (and almost certainly other) processors.



Security News

The BleedingBit Bluetooth flaws
https://armis.com/bleedingbit/

It's got a catchy name and a nice drippy bleeding logo... so what is it?

Armis security has discovered and responsibly reported two chip-level Bluetooth Low Energy vulnerabilities affecting some enterprise-class WiFi routers. Consequently, they are not BLE protocol definition flaws, they are specific implementation flaw.

Armis wrote the following:

> *Armis has identified two chip-level vulnerabilities impacting access points and potentially other unmanaged devices. Dubbed "BLEEDINGBIT," they are two critical vulnerabilities related to the use of BLE (Bluetooth Low Energy) chips made by Texas Instruments (TI). The chips are embedded in, among other devices, certain access points that deliver Wi-Fi to enterprise networks manufactured by Cisco, Meraki and Aruba. These are the leaders in networking, accounting for nearly 70% of the [enterprise] market.*
>
> *These proximity-based vulnerabilities allow an unauthenticated attacker to break into enterprise networks undetected. Once an attacker takes control over an access point, he can move laterally between network segments to create a bridge between them — effectively breaking network segmentation.*
>
> *Armis has reported the issues to TI and the affected vendors above. We are also working with additional vendors of various connected devices to ascertain whether they, too, are affected by the BLEEDINGBIT vulnerabilities.*

Armis will be presenting their work at next month's BlackHat Europe December 3-6 under the title: "BLEEDINGBIT: Your APs Belong to Us"...

> *Enterprise Wi-Fi access points are the gateway to the network of any business today. They connect both our corporate endpoints, as well as various unmanaged IoT devices that have become an essential part of our networks. We trust access points to create segments in our networks for these devices and offer separate guest networks to visitors. Modern access points implement novel functionalities, which allow detection of various interferences, and offer mobility and location services. However, these new features also introduce risks that create a new network attack surface.*
>
> *In this talk, we will demonstrate two new zero-day vulnerabilities in prominent chips used by the majority of wireless access points, that allow an unauthenticated attacker to penetrate an enterprise network undetected. After compromising the access point, an attacker has full control and can read traffic passing through the access point, distribute malware, and even move laterally between network segments.*

Disclosure: Armis contacted Texas Instruments on June 20, 2018. Through our discussions, it was discovered that TI was familiar with the bug causing the vulnerability, and issued a fix in BLE-STACK v2.2.2. However, Armis identified it as a security issue. Once notified, the companies worked together to issue the appropriate updates to the patch, and coordinate the announcements. Cisco was notified on July 24, 2018 of the issue.

Okay… so TI knew there was a problem but apparently they weren't very concerned about it. So wait until you hear how capable security researchers are able to turn a bug into a truly significant vulnerability!:

BLEEDINGBIT RCE vulnerability (CVE-2018-16986)
The first BLEEDINGBIT RCE (Remote Code Execution) vulnerability resides in a TI chip embedded in many devices. Our research focused on access points. The vulnerability can be exploited by an attacker in the vicinity of the affected device, provided its BLE is turned on, without any other prerequisites or knowledge about the device. First, the attacker sends multiple benign BLE broadcast messages, called "advertising packets," which will be stored on the memory of the vulnerable BLE chip in targeted device. While the packets are not harmful, they contain code that will be invoked by the attacker later on. This activity will be undetected by traditional security solutions.

Next, the attacker sends the overflow packet, which is a standard advertising packet with a subtle alteration – a specific bit in its header turned ON instead of off. This bit causes the chip to allocate the information from the packet a much larger space than it really needs, triggering an overflow of critical memory in the process. The leaked memory contains function pointers – memory that points to specific code segments, which the attacker can leverage to point to the code s/he sent to the vulnerable chip in the previous stage of the attack.

At this point, the attacker can run malicious code on the targeted device, and install a backdoor on the vulnerable chip, which will await further commands transmitted over BLE. The attacker can also change the behavior of the BLE chip and attack the main processor of the device, gaining full control over it. In the case of an access point, once the attacker gained control he can reach all networks served by it, regardless of any network segmentation. Furthermore, the attacker can use the device in his control to spread laterally to any other device in its vicinity, launching a truly airborne attack.

BLEEDINGBIT OAD RCE vulnerability (CVE-2018-7080)
The second BLEEDINGBIT vulnerability was specific to the Aruba Access Point Series 300, and its use of the OAD (Over the Air firmware Download) feature from TI. This issue is technically a backdoor in BLE chips that was designed to allow firmware updates. The OAD feature is often used as a development tool, but is active in some production access points. It can allow a nearby attacker to access and install a completely new and different version of the firmware — effectively rewriting the operating system of the BLE chip, if not implemented correctly by the manufacturer.

By default, the OAD feature is not automatically configured to address secure firmware updates. It allows a simple update mechanism of the firmware running on the BLE chip over a GATT [Generic Attributes] transaction. In the case of Aruba's access points, a hardcoded password was

added (that is identical across all Aruba APs that support BLE) to prevent the OAD feature of being easily abused by attackers.

However, an attacker who acquired the password by sniffing a legitimate update or by reverse-engineering Aruba's BLE firmware can connect to the BLE chip on a vulnerable access point and upload a malicious firmware containing the attacker's own code, effectively allowing a completely rewrite its operating system, thereby gaining full control over it. From this point, the malicious potential is identical to that achieved by the first vulnerability.

Remediation:
Firmware updates were provided by TI to their OEM device manufacturers. So if you're a user of the affected devices you'll be wanting to jump on this immediately.

The OAD (Over the Air firmware Download) can be and should be turned off immediately -- perhaps after performing one final Over-the-Air firmware update!

JavaScript: No longer optional with Google
https://security.googleblog.com/2018/10/announcing-some-security-treats-to.html

Last Wednesday on Halloween, Google product manager Jonathan Skelker began his blog posting by writing:

> *"It's Halloween, and the last day of Cybersecurity Awareness Month, so we're celebrating these occasions with security improvements across your account journey: before you sign in, as soon as you've entered your account, when you share information with other apps and sites, and the rare event in which your account is compromised."*
>
> *[...] When your username and password are entered on Google's sign-in page, we'll run a risk assessment and only allow the sign-in if nothing looks suspicious. We're always working to improve this analysis, and we'll now require that JavaScript is enabled on the Google sign-in page, without which we can't run this assessment.*
>
> *Chances are, JavaScript is already enabled in your browser; it helps power lots of the websites people use everyday. But, because it may save bandwidth or help pages load more quickly, a tiny minority of our users (0.1%) choose to keep it off. This might make sense if you are reading static content, but we recommend that you keep Javascript on while signing into your Google Account so we can better protect you. You can read more about how to enable JavaScript here.*

Note: "Recommending that Javascript be kept on" is decidedly different from "we'll now require that JavaScript is enabled." It's clear that this is going to be a requirement, not a "recommendation." And the text of our picture of the week should remove any doubt.

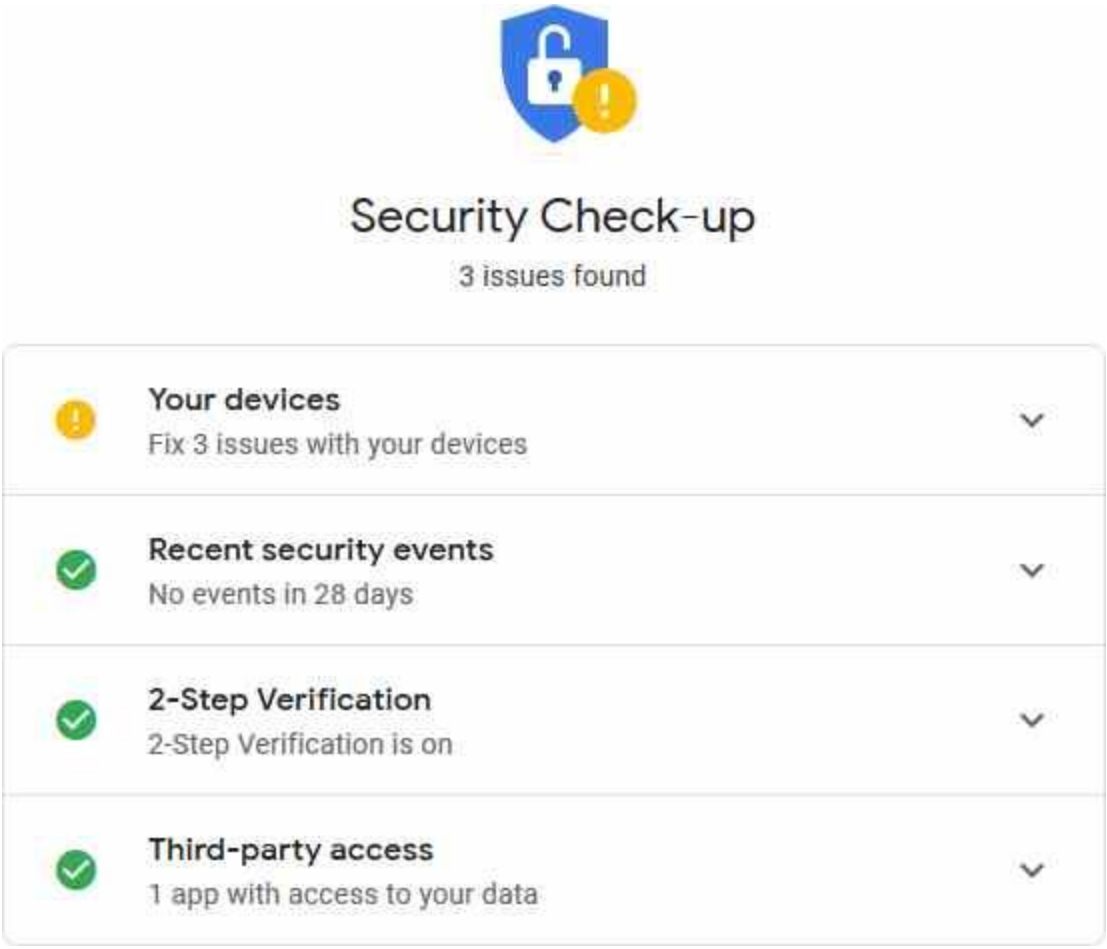
Additionally, in this posting Jonathan wrote under the banner of "Keeping your Google Account secure while you're signed in":

Last year, we launched a major update to the Security Checkup that upgraded it from the same checklist for everyone, to a smarter tool that automatically provides personalized guidance for improving the security of your Google Account. We're adding to this advice all the time [and we recently] introduced better protection against harmful apps based on recommendations from Google Play Protect, as well as the ability to remove your account from any devices you no longer use.

So, in other words, Google Play Protect now provides feedback to the devices' Security Checkup and will warn the user when they have anything installed that Google no longer feels is behaving honorably. However, it appears that users will need to deliberately access the Security Checkup. So now that this facility is in place, our listeners should take a little visit over to the Security Checkup

https://myaccount.google.com/security-checkup

I took my own advice….



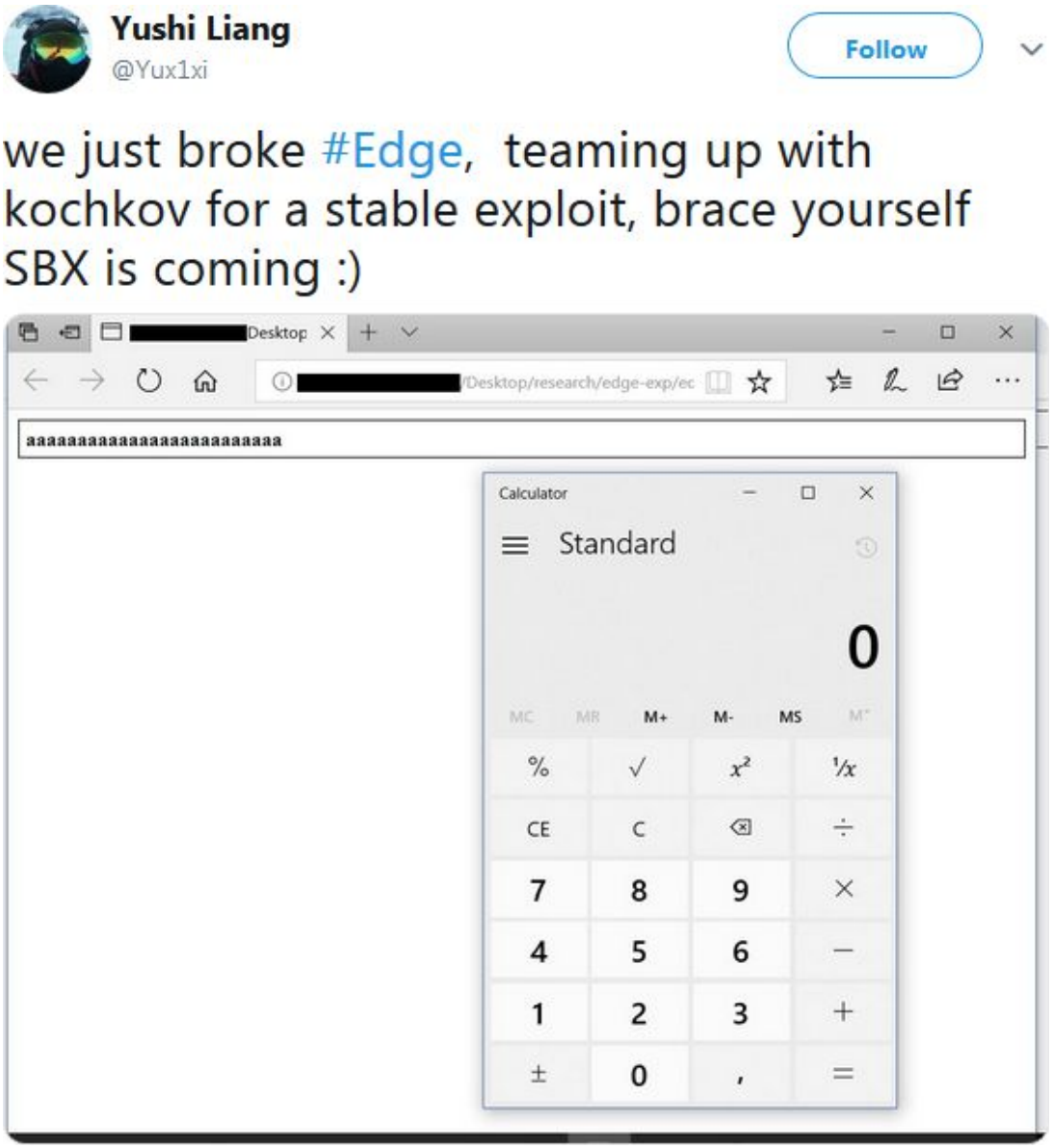
It found three devices that I hadn't used in a LONG time and recommended that I remove my Google account from them.

Google has also expanded their notification of apps having access to Gmail data or Google Contacts to include warnings when ANY Google Account data is being shared with websites or local device apps. And remember that the "Security Checkup" can always be used to audit the sites and apps which currently have access.

Google is also claiming that they'll be involved in helping to recover from an account breach... and from its consequences:

Under: *"Helping you get back to the beginning if you run into trouble"*
In the rare event that your account is compromised, our priority is to help get you back to safety as quickly as possible. We've introduced a new, step-by-step process within your Google Account that we will automatically trigger if we detect potential unauthorized activity.
We'll help you:

- Verify critical security settings to help ensure your account isn't vulnerable to additional attacks and that someone can't access it via other means, like a recovery phone number or email address.
- Secure your other accounts because your Google Account might be a gateway to accounts on other services and a hijacking can leave those vulnerable as well.
- Check financial activity to see if any payment methods connected to your account, like a credit card or Google Pay, were abused.
- Review content and files to see if any of your Gmail or Drive data was accessed or mis-used.



Yushi is a well known hacker/developer with some track record, so this tweeted announcement of a discovery of a 0-day exploit of Edge with SBX (sandbox escape) is probably authentic.

Previously, Yushi developed a remote code exploit for Firefox which required the use of a three-bug exploit chain and on the Chromium browser he was able to achieve code execution, though without a sandbox escape.

Upon seeing his latest claim, the guys at Bleeping Computer reached out and received a video demonstrating the Edge breach. In the video, Microsoft's Edge browser is made to lanch an instance of Firefox (which should definitely not be possible) and have Firefox load the download page for Google's Chrome browser.

Yoshi will be attending the forthcoming Pwn2Own Mobile hacking competition in Tokyo next Tuesday and Wednesday where he can be expected to show off some of his other exploits and taking home some prizes. His recent tweets have revealed exploits against desktop web browsers, but next week's mobile Pwn2Own is focused upon smartphones and IoT devices including phones from Google, Samsung, Apple and Huawei and IoT devices including:

- Apple Watch Series 3, Amazon Echo (2nd Generation), Google Home, Nest Cam IQ Indoor
- Amazon Cloud Cam Security Camera

More than $500,000 USD in cash and prizes are available to researchers with ten different devices available as targets in a total of five different categories. Since Zerodium now pays out $50,000 for a qualifying 0-day remote code execution exploit in Edge, and doubles that when sandbox escaping is achieved, it's nice to see responsible hackers participating in these Whitehat events where their exploits are not being sold for resale to the underworld.

Windows Defender gets to play in its own Sandbox
https://cloudblogs.microsoft.com/microsoftsecure/2018/10/26/windows-defender-antivirus-can-now-run-in-a-sandbox/

This is a big deal, and it was very difficult to pull off. Although Microsoft isn't saying anything for antitrust / anti-competitive reasons… doing this required deep and careful plumbing into Windows which no other Windows add-on A/V will be able to truly duplicate. This is no fault of Microsoft's, it's just the reality of the nature of the problem that any contemporary A/V must face. The era of the 3rd-party A/V is likely coming to an end.

What's the big deal?

As we've discussed here a number of times in the past, threat modelling is all about understanding the "attack surface." A web browser presents a large and rich attack surface for our modern PCs. But another "attack surface" is untrained employees who freely click on anything that looks enticing. People are a widely exploited attack surface too.

But perhaps the largest attack surface there is, is today's anti-virus subsystem whose job it is to proactively examine **everything** coming into the machine it has been tasked to protect. So that's things users click on in eMail and in their web browser and that they receive in mail and download from their browser.

A/V presents one of the richest attack surfaces because:

- It's an interpreter which examines and attempts to understand what it's seeing.
- As we know, interpreters are among the most difficult technologies to secure.
- An A/V process must run with full system privileges since it needs to have total visibility into every nook and cranny of the system's permanent storage, RAM, and the network.

This means that A/V is a huge target of opportunity, because it is both inherently prone to exploitation because it must be **so careful** when it looks at potential malware, and because to do its job it **must** execute with full access to the entire system.

Microsoft's blog posting further details the very many challenges the Defender engineers faced. But suffice to say that it was a truly difficult job, over a long period of re-engineering… and reading it I realized that not only was it extremely difficult, but that it inherently required inside access to Windows which no external player could ever have.

And it's a big deal. Tavis Ormandy, the prolific and frequently showering researcher with Google Project Zero who has previously discovered and disclosed several of these types of flaws in the past year, lauded the Microsoft's effort on Twitter, saying it was "game-changing."

All that said, Microsoft is proceeding with caution, so Sandboxing is not yet enabled by default. Microsoft wrote: "We're in the process of gradually enabling this capability for Windows insiders and continuously analyzing feedback to refine the implementation." But anyone with Windows 10 version 1703 or later -- last year's Fall Creator's Update -- can enable Defender's sandboxing for themselves:

To enable Windows Defender's Sandboxing:

- Open Start and Search for "CMD" or "Command Prompt"
- Right Click on it and select "Run as administrator."
- Type: "setx /M MP_FORCE_USE_SANDBOX 1" and then press ENTER.
- Then restart your computer.

However! -- Restart and do not Shutdown Windows. There's a bug which Microsoft has acknowledged and will eventually fix such that sandboxing changes will ONLY be saved if the system is restarted and not shutdown and restarted.

Once the sandboxing is enabled, and the system has been restarted without shutdown, its presence can be verified by using the free SysInternals "Process Explorer" utility.

The traditional antimalware service is "MsMpEng.exe" and a new child process named "MsMpEngCP.exe" will be attached to it, running underneath and indented. If you see "MsMpEngCP.exe" your Windows Defender has been safely sandboxed. "CP" stands for Content Process, which is Microsoft's formal name for sandboxed processes.

I tried and verified all that. It all works great! :)

And speaking of SysInternals…
Two guys... Mark Russinovich and Bryce Cogswell first created "Winternals Software LP" and the SysInternals website 22 years ago, way back in 1996.

It was a huge hit with Windows techies everywhere, since "SysInternals" offer a unique and unmatched suite of 100% free system-related tools and utilities for determining much more about what was going on inside Windows beneath the covers than was available elsewhere.

Many of their tools became classics and I knew of many people who routinely made snapshots of the entire collection just in case anything might happen to make them suddenly unavailable.

In addition to "Process Explorer" -- an advanced version of Windows Task Manager -- "Autoruns" was a favorite of everyone's since it provided the best comprehensive view into what was being run at system startup or logon. And, of course, their "RootKitRevealer" cleverly detected discrepancies between the Windows Registry and file system API which might suggest the presence of a rootkit. It was that THAT utility which first alerted users that Sony BMG was installing a rootkit -- which could be abused for malicious purposes to hide OTHER software from view -- into the Windows systems of everyone who used those Sony products.

What DID finally happen, to everyone's horror, was the announcement that Microsoft was acquiring SysInternals and hiring the boys. Let me tell you that MANY many snapshots were made of their entire tool collection following that announcement. But the years went by, and their tools kept growing and maturing and new one's appeared. Just as with Microsoft's recent purchase of Github, it all turned out for the better.

And this is all relevant today because we have recently learned that Microsoft is in the process of porting the famous SysInternals tools to Linux! The ProcDump utility has already been ported, and ProcMon is next… with more tools to follow.

https://github.com/Microsoft/ProcDump-for-Linux

ProcDump:
ProcDump is a Linux reimagining of the classic ProcDump tool from the Sysinternals suite of tools for Windows. ProcDump provides a convenient way for Linux developers to create core dumps of their application based on performance triggers.

According to Mario Hewardt, Principal Program Manager for Azure Diagnostics at Microsoft, these first ports are part of the company's larger plan to make the Sysinternals package available for Linux users in the future.

The "CAPTCHA" further evolved
https://webmasters.googleblog.com/2018/10/introducing-recaptcha-v3-new-way-to.html

From Monday before last (which I missed because I had prepared the podcast the Sunday before and was off in reunion mode)

CAPTCHA is the acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart" And we have examined and explored the challenge of the CAPTCHA any number of times through the years here.

Now Google, the parent of the "reCaptcha" has delivered the third major version of their solution. They announced it to webmasters, so this is aimed not at end users but at website designers, saying:

Today, we're excited to introduce reCAPTCHA v3, our newest API that helps you detect abusive traffic on your website without user interaction. Instead of showing a CAPTCHA challenge, reCAPTCHA v3 returns a score so you can choose the most appropriate action for your website.

A Frictionless User Experience
Over the last decade, reCAPTCHA has continuously evolved its technology.

In reCAPTCHA v1, every user was asked to pass a challenge by reading distorted text and typing into a box.

To improve both user experience and security, we introduced reCAPTCHA v2 and began to use many other signals to determine whether a request came from a human or bot. This enabled reCAPTCHA challenges to move from a dominant to a secondary role in detecting abuse, letting about half of users pass with a single click.

(That was the famous, if somewhat confounding "I'm not a Robot" checkbox assertion.)

Now with reCAPTCHA v3, we are fundamentally changing how sites can test for human vs. bot activities by returning a score to tell you how suspicious an interaction is and eliminating the need to interrupt users with challenges at all. reCAPTCHA v3 runs adaptive risk analysis in the background to alert you of suspicious traffic [to your website] while letting your human users enjoy a frictionless experience on your site.

More Accurate Bot Detection with "Actions"
In reCAPTCHA v3, we are introducing a new concept called “Action”—a tag that you can use to define the key steps of your user journey and enable reCAPTCHA to run its risk analysis in context. Since reCAPTCHA v3 doesn't interrupt users, we recommend adding reCAPTCHA v3 to multiple pages. In this way, the reCAPTCHA adaptive risk analysis engine can identify the pattern of attackers more accurately by looking at the activities across different pages on your website. In the reCAPTCHA admin console, you can get a full overview of reCAPTCHA score distribution and a breakdown for the stats of the top 10 actions on your site, to help you identify which exact pages are being targeted by bots and how suspicious the traffic was on those pages.

So, as with many of the services Google offers, there’s a privacy & tracking tradeoff here. Just as the addition of Google Analytics provides the webmaster with great quantities of doubtless useful information presented through a beautiful UI, those same great quantities of useful information also flows back to Google. And it’s clear that a similar transaction is going on here. Google makes clear that they want their new “adaptive risk analysis” reCAPTCHA to be sprinkled liberally throughout a website to give it greater visibility into user behavior. But that also gives Google that much greater visibility into your website’s visitor behavior. I’m not suggesting that there’s anything wrong with that, but it might be worth keeping in mind.

Last Friday was the 30th anniversary of the Internet's first worm: The Morris Worm
Steven J. Vaughan-Nichols writing for ZDNet...
https://www.zdnet.com/article/the-day-computer-security-turned-real-the-morris-worm-turns-30/

On Nov. 2, 1988, [Steven writes] I was working at NASA's Goddard Space Flight Center in the data communications branch. Everything was fine. Then, our internet servers running SunOS and VAX/BSD Unix slowed to a stop. It was a bad day.

We didn't know it yet, but we were fighting the Morris Internet Worm. Before the patch was out, 24 hours later, 10 percent of the internet was down, and the rest of the network had slowed to a crawl. We were not only facing the first major worm attack, we were seeing the first distributed denial-of-service (DDoS) attack.

Unlike the hundreds of thousands of hackers that would follow, Robert Tappan Morris, then a graduate student at Cornell, wasn't trying to "attack" the internet's computers. He thought his little experiment would spread far more slowly and not cause any real problems. He was wrong.

Well, that's what he said afterward. I'm also not at all certain that that was the case.

Consider, the Morris worm had three attack vectors: sendmail, fingerd, and rsh/rexec. It also used one of the now-classic attack methods: Stack overflow in its attack.

It was also one of the first attack programs to use what we'd call a dictionary attack with its list of popular passwords. The passwords and other strings hid in the Worm's binary by XORing, a simple encryption method.

Morris also tried to hide his tracks. He started the worm from an MIT computer. It hid its files by unlinking them after trying to infect as many other servers as possible.

Even without a malicious payload, the Worm did serious damage. Infected systems quickly did nothing but trying to spread the worm, thus slowing them down to a crawl. Some, most of them running SunOS, a Unix variant and the ancestor of Solaris, crashed under the load.

In the meantime, Morris, who included code to keep the worm from spreading too fast, had realized he was no longer in control. Morris called a friend -- who subsequently said Morris "seemed preoccupied and appeared to believe that he had made a 'colossal' mistake.'"

He had indeed. Thanks to efforts led by Eugene "Spaf" Spafford, then an assistant professor of computer science at Purdue University and current editor-in-chief of Computers and Security, the Worm was conquered.

Before the Worm was finished, it successfully attacked about 6,000 of the 1988 internet's 60,000 servers. In the aftermath, DARPA created the first CERT/CC (Computer Emergency Response Team/Coordination Center) at Carnegie Mellon University to deal with future security attacks.

But the Worm's biggest legacy to date was that it started a wave after wave of computer and internet attacks. If Robert Morris hadn't done it, someone else would have. But, regardless, today we live in a world where a day doesn't go by without a serious attack.

CommonRansom Ransomware Demands RDP Access to Decrypt Files
https://www.bleepingcomputer.com/news/security/commonransom-ransomware-demands-rdp-access-to-decrypt-files/

BleepingComputer's Lawrence Abrams reported on a crazy new ransomware that should only be replied to as a last resort and if then only after taking extreme precautions:

The ransom note reads:

Hello dear friend.
Your files were encrypted!
You have only 12 hours to decrypt it.
In case of no answer our team will delete your decryption password.
Write back to our e-mail: old@nuke.africa.

In your message you have to write:

1. This ID-[VICTIM_ID]
2. [IP_ADDRESS]:PORT(rdp) of infected machine
3. Username:Password with admin rights
4. Time when you have paid 0.1 btc to this bitcoin wallet: 35M1ZJhTaTi4iduUfZeNA75iByjoQ9ibgF

After payment our team will decrypt your files immediately.
(The note then proceeds to explain how to obtain bitcoins.)

Now… who the heck is going to give bad guys an RDP connection to an encrypted system?

As we know, since Windows workstations only allow one interactive user at a time, the screen will go blank and the victim will have NO IDEA what the bad guys are doing.

IF you were to be infected by this nightmare, the best advice would be not to get yourself infected in the first place. But if that ship have sailed and you have no backups and really really no other choice…

Then take EVERYTHING -- absolutely everything -- else off of your network. Perhaps change the WiFi password so that everything disconnects. And arrange to ONLY have the one infected machine on the network.

Once you regain control of your machine immediately physically remove its drive and set it aside. Install a new drive and reinstall Windows again from scratch. If you have an older backup you might then restore from that backup, but NEVER trust that removed hard drive again. You can NEVER execute anything from it. You should treat it as a contaminated drive from which you only ever manually extract those DATA files that you absolutely must have. Contaminated software on that drive is not mystical. It cannot jump from the drive. But you must be certain to never run anything from that drive. ONLY its DATA is safe.

The sounds of silence:

Page 13 of Apple T2 Security Chip Security Overview / October 2018

"Hardware microphone disconnect"
All Mac portables with the Apple T2

Security Chip feature: A hardware disconnect that ensures that the microphone is disabled whenever the lid is closed. This disconnect is implemented in hardware alone, and therefore prevents any software, even with root or kernel privileges in macOS, and even the software on the T2 chip, from engaging the microphone when the lid is closed. (The camera is not disconnected in hardware because its field of view is completely obstructed with the lid closed.)

SpinRite

Sunday, 9:07pm
Jonathan Hellewell @jrobertbooks

Hey Steve,

I have been a fan of Security Now since I found it a few years ago, during the Snowden saga. I get a kick out of it when certain technologies or hardware comes up in other media. While the news types hype tech stories beyond credulity, I always wonder what your take on those issues is going to be. Bloomberg's apparent if unintentional hit piece on Supermicro comes to mind as a recent example.

I want to thank you for Spinrite. I recently had my old winblows box die. The computer was no real loss, but a single folder was critical. I'm a self-published author and the folder with all of my research, notes, cover images and drafts was on that computer. That system was setup for weekly backups, but it died on day six. I was left banging my head on the desk… I'd written 10,000+ plus words on the second book of my series that week and trying to redo it from memory really wasn't something I wanted to do. Spinrite came through for me. Twelve hours later, I was able to get the drive to mount and I pulled that folder off the drive without any extra problems.

So, thanks for saving a bit of my sanity. If you find yourself looking for some sci-fi, maybe you could give the Emerging Ascendancy series a look at jrobertbooks.com. I'd be more than happy to send you a copy to say thanks. (ebook or print) I'm rather curious to see if you'd like it.

Thanks again,
Jonathan

Closing The Loop

Triple Stuf (@TripleStufOreo)

Hi Steve,

I have a question regarding your vending machine puzzle. I came up with a similar solution, but I don't understand some of the choices you made. Wouldn't it be even simpler and more secure to have the following differences?

What if the vending did not have any HSM? All it has is the server public key, a bunch of nonces and a vending machine identifier. When someone wants to buy something via the app, the app sends the request (containing the nonce, vending machine identifier and product of choice) to the server. The server then of course checks the balance, deducts the correct amount and sends back a signed version of the request, therefore acknowledging the successful transaction. The app gives the signed request to the vending machine which it can check.

Advantages:

• Simpler system with less hardware
• Vending machine doesn't have any secrets to keep (even though this shouldn't be a problem for an HSM)

Note: If you wanted to send the inventory securely, this data could also be put in the request that has to come back signed.

Please explain to me what I'm missing. Thanks for the great show every week!

If you would like to use my question in SN, please feel free to do so.

Kind regards,
Rein from the Netherlands

Mike Calandra (@CalandraMike)

Another thought, I heard you say on the last podcast that someday you would consider switching from firefox to chrome, despite the fact that google uses your browsing history a primary source of revenue and hence it is less private. Your show is on privacy and security, how do you reconcile this? It seems that chrome would have to have significant advantage over firefox for you and I wonder what that would be? I've always been a firefox use. Big fan of the podcast, thank you.

Reminders to self: uBlock Origin, Disconnect.me, etc.

“PortSmash”

So… It doesn’t have quite the ring to it as “BleedingBit”, and if it had been my discovery I might have been tempted to name this next vulnerability “YAIPE” for Yet Another Intel Processor Exploit. But in any event, its discoverers have chosen to name it “PortSmash.”

The short version of this poses the question: Is Intel’s Hyper-Threading dead? (Remember that OpenBSD, out of a characteristic abundance of caution, completely disabled its OS’s use of hyper-threading earlier this year when the Spectre and Meltdown disasters began to unfold.) Near the start of last month, PC World’s executive editor headlined a story: “Intel's 9th-gen Core i7-9700K abandons Hyper-Threading: What it could mean for performance. Intel's 9th-gen gives the Core i7 a demotion in threads, but a promotion in actual cores.”

PC World explains:

Intel first introduced Hyper-Threading on consumer CPUs with the Northwood-based Pentium 4 in 2002. It works by splitting a single physical core into a two logical cores. Since most compute threads don’t consume 100 percent of a CPU’s resources, Hyper-Threading lets the unused resources do work as well. Hyper-Threading, of course, is Intel’s fancy pants name for simultaneous multi-threading (SMT), which AMD also began employing with its Ryzen chips.

Although Hyper-Threading’s performance boost has been around for 16 years, it hasn’t always been tapped into. No Core 2 CPUs ever used the feature, for example, and Intel’s Atom CPUs have had it off and on.

That was back in October; and it’s clear that Intel has long been juggling -- and judging -- the tradeoffs between complexity, power consumption, thread count and hyperthread utilization, and has not always come down with a purely pro-hyperthreading position.

So here we are a month later, and something bad just happened to hyper-threading:

The proof of concept code, called PortSmash, comes from researchers at technical universities in Finland and Havana, Cuba. PostSmash is a new (it’s not another variant of Spectre or Meltdown) side-channel attack and exploit with working proof of concept code posted on Github.

https://github.com/bbbrumley/portsmash

The PoC successfully steals a TLS session’s secret key under OpenSSL across the “threading” boundary. In other words, across two logically separate processes which **should** be completely isolated from one another. The researchers have convincingly demonstrated that cross-hyper-thread information leakage **can** and **does** occur.

The researchers write: “A CPU featuring SMT (e.g. Hyper-Threading) is the only requirement. This exploit code should work out of the box on Skylake and Kaby Lake. For other SMT architectures, customizing the strategies and/or waiting times in “spy” is likely needed.
They named it “PortSmash” because, at the micro-architectural level, each physical hardware

core is subdivided into a number of separable regions, called “ports”, each of which perform different tasks within the processor and are, by being split up into separated ports, able to operate separately. In other words, by granularizing a single core, its different ports which no one thread is using all at once, can be doing work for another thread at the same time.

And, by now, after nearly a year of Spectre and Meltdown revelations, we know where this is headed, right? … “PortSmash” uses subtle instruction timing variations which allow it to detect the contention which inevitably exists when two logically separate threads of execution are sharing a single core’s hardware.

PortSmash’s attacking thread repeatedly executes instructions until the CPU’s hyperthread scheduler stops it running and hands the port over to the other thread. By measuring the time in between its own instructions running on that port, it can measure the time that the other thread takes to process its own instructions. And this, in turn, can help it derive another program’s secrets over time.

For their part, Intel wote:

> *This issue is not reliant on speculative execution, and is therefore unrelated to Spectre, Meltdown or L1 Terminal Fault. We expect that it is not unique to Intel platforms. Research on side-channel analysis methods often focuses on manipulating and measuring the characteristics, such as timing, of shared hardware resources. Software or software libraries can be protected against such issues by employing side channel safe development practices. Protecting our customers’ data and ensuring the security of our products is a top priority for Intel and we will continue to work with customers, partners and researchers to understand and mitigate any vulnerabilities that are identified.*

OpenSSL has already been updated to mitigate the effects of this attack so that updated versions of OpenSSL are no longer vulnerable.
So what about hyper-threading?

Hyper-threading itself is useful since it does squeeze an additional ~10-20% out of a single core’s microcode execution units by keeping them more often busy by giving them more work to do. So despite its potential for abuse, once the potential downsides are clearly understood, hyper-threading **can** be used safely. Therefore, I doubt that we’ll be seeing it disappearing altogether.

In the first place, let’s not lose sight of the fact that the **only** place where danger is present is in a scenario where there **could** be a hostile instrumented thread simultaneously running on the same core as a thread which is performing security-sensitive work. That’s the attack model -- a hostile thread simultaneously sharing a core with a security-sensitive thread.

So the maximum danger really only exists in shared host virtual machine scenarios where unknown and untrusted processes are all cross-sharing the virtualized hardware at the same time. No one is denying the danger there. It’s real… and no we know more clearly how real it is.

But an enterprise web or database server is only running its own server code. It’s not running random unknown foreign processes. And the same is generally (though perhaps a bit less

explicitly) true for experprise and personal workstations. As we've often observed, if you have something bad in your machine which **could** be leveraging Spectre, Meltdown or now PortSmash you've already lost the battle.

But also...

Many of today's processes are internally multi-threaded, meaning that within a single process many things are going on at the same time. So the thread schedulers in our operating systems **can be** usefully and relatively easily updated so that no single core is simultaneously shared among multiple processes. In other words, hyper-threading **can** be safely employed by multiple threads within a single process, since all of a processes threads are on the same team. The "PortSmash" danger is only present when separate processes are simultaneously sharing a single physical core.

So, with a bit of tweaking of the schedulers of the operating systems running within shared hosting VM systems -- even in the presence of hostile processes -- the benefits conferred by hyper-threading can be obtained without risk.

And coming up Next Week:

A deep dive into Self-Encrypting SSD drives.
Just how safe is the implementation of their internal encryption.
(Hint: It isn't!)

~30~