Security Now! #946 - 10-31-23

Citrix Bleed

This week on Security Now!

What caused last week's connection interruption? Is it possible to create and maintain an Internet whitelist? What's the latest on LastPass vault decryptions? How do you know of a remote correspondent adds a new device to their Apple account that it's really them? Might there be more life left in Windows 10 than we thought? What's foremost in the minds of today's bug bounty hunters? What new free and open source utility has CISA released? Could it be that SpinRite 6.1 is finished? Is TLS 1.2 ready for retirement? And what about IPv4? How can open source projects get their code signed? And then we're going to take a really interesting deep dive into the Internet's latest mass-casualty disaster.

What’s the plan here?

Miscellany

Router was rebooting

After the connection interruptions during last week's podcast, I got serious about tracking down the cause of the occasional interruptions that my network had been suffering. When I'm focused on work I'll ignore pretty much anything else. But having the podcast interrupted is not okay. By being a lot more proactive when this occurred I was able to track the cause of the trouble down to occasional spontaneous crashes and reboots of my little Netgate SG-1100 router which runs pfSense and sits just inboard of my cable modem. It enforces my network's segmentation, creates separate WiFi domains, and does a bunch more that I've talked about in the past.

The first thing to check when something like this is happening is the system's power supply. The little "wall warts" that ship with consumer grade equipment are often the source of trouble. If the power supply is getting tired it could be the source of the trouble. Fortunately, the little SG-1100 uses a standard 12 volt DC supply with the standard DC barrel connector. So I had plenty of swap-in replacements at hand. Since I exchanged its power supply I have had zero reboots. I still have my fingers crossed, but since it was previously happening sometimes several times a day, I'm hopeful that the culprit has been identified and eliminated.

David Redekop -&- AdamNetworks

Leo, you'll remember another of this podcast's very early sponsors: the "Nerds on Site" guys based in Canada and with their "Nerds" spread far and wide to help non-computer savvy users and enterprises with their computer problems. They were early fans of the podcast and I keynoted their annual gathering once or twice. I also first met them when I was in Toronto as a guest on your Call For Help show for Rogers Cable. So we're talking 18 years ago in those very early audio-only days of the podcast when we used to record in your hotel room.

Anyway, since then I've stayed loosely in touch with David Redekop, who was one of the principals and I've been vaguely aware of what he's been doing. He was passing through town last week, so we rendezvoused at Starbucks to catch up. Now, this was over my second 5-shot latte of the morning, so my brain was fully charged. David took about an hour to walk me step-by-step through the careful and deliberate design of the network security solution he and his team of techies at AdamNetworks have designed and which is currently in use actively protecting around two million user and device endpoints around the world. So this solution is not just theory, and a surprisingly good idea; it's actually in place and working. And once I'd heard what this group has managed to pull, off that didn't surprise me because it felt like the result of a decade or more of iterating on a concept to maturity. You start with an idea, you implement a first pass and put it out there; you remain involved and connected; you learn what does and doesn't work; then you iterate and push out an improved solution. You add new features to address new requirements, you test them and adjust them until they are working correctly.

I've often noted on this podcast that I cannot imagine the challenge of keeping an enterprise safe while it's connected to today's Internet. And in fact, today's "Citrix Bleed" topic makes me shudder a bit because even if everything was done right it's still possible to be taken down. I am **so** glad that keeping a highly-connected sprawling enterprise safe is not my job. But it does still need to be done, and these AdamNetworks guys have not only imagined the challenge of keeping an enterprise safe, from what was explained to me last Friday – and I think I pretty

much understand the way it works, at least in broad strokes – they've not only imagined the challenge, they've risen to it; and if I had an enterprise network which I needed to protect today, based upon what I understand now, I would not want to be without this solution. I'm thinking of reaching out to Father Robert to make him aware of this because it would be very cool to have The Vatican this well protected.

Nobody today runs their firewall where by default everything is allowed through except for traffic that's known to be malicious. In the dim dark ages of the Internet we once blocked known exploitable ports. We don't do that anymore. But blocking known bad domains or scanning communications traffic for known malicious content **is** what's being done today and it's the modern day equivalent of a default "allow all" rule at the bottom of a block list. We do it because we think there's no alternative. Last Friday, the AdamNetworks guys showed me a better way.

What these guys have done is to not worry about inspecting and interpreting the contents of enterprise user communications. As I understand it, they do not do any deep packet inspection; so no one's communications contents privacy is ever compromised. What they've focused on, specialized in, and made work, is only allowing connections to known safe and benign entities. If it were possible to do this perfectly, phishing attacks would be ineffective because a user clicking on a phishing link would be unable to obtain and malicious content, and even if someone were to inadvertently bring some malware into the enterprise that infected it from inside, if that malware was then unable to connect to its command and control servers, it would be effectively inert.

So imagine that you start with very strong DNS filtering and require all clients on the network to use this DNS. They've also extended this through, and enforced it for, enterprise controlled smartphone and other mobile clients. The thing that sets this apart from other solutions is that this is not the typical black listing DNS, which selectively blocks known bad domains. AdamNetworks has managed to build a practical **white listing DNS** which by default blocks everything unknown and only permits DNS resolution for known safe domains. DNS is resolved on premises, so privacy is preserved, but it's also informed by a shared central manager in the cloud so that as new domains come online and are cleared, the master whitelist can be updated. If a user attempts to connect to an unlisted DNS domain they receive an intercept page (the way this is done is quite clever) which they must manually accept and approve. But before that is allowed, 18 other public sources of DNS filtering from other providers are consulted to make sure the unknown domain is not known to be malicious by anyone. There's technology in the pop-up intercept page which prevents malware from simulating the human's permission.

And speaking of malware, it turns out that modern malware is hip to having its DNS blocked. So there **is** some malware that has static IP addresses burned-in; it doesn't do any DNS, it just connects directly to its command and control. So this introduces the outbound firewall component of this solution which they call "Don't Talk To Strangers" (DTTS): No outbound connection is allowed to any IP that wasn't first looked up with the system's DNS resolver. So by linking DNS resolution to a dynamic outbound firewall, DNS, which has already been made very strong, becomes the gatekeeper. But then it turns out that Facebook, for example, has some apps that also don't do DNS. Facebook's apps are benign and they know the fixed IP addresses they want to connect to back at the mothership. So it's necessary for the system to incorporate some white listing for known benign blocks of destination IPs. So that's in there, too.

What I've described just scratches the surface of the enterprise security solution these guys have developed and I'm sure that our techie listeners are saying "but but but but..." to which all I can say is that they actually have this thing working. They demonstrated it to me and, as I said, more than two million user and device endpoints are currently under this solution's protection.

They have a full mature management UI, scheduling, per-user permissions and everything else needed to implement a practical enterprise class solution. I asked David how much trouble there was with unknown yet benign DNS and he acknowledged that it was there, but that they had reduced the level to such a degree that it was effectively negligible. I recall that he said 5%, but I don't remember 5% of what. My reaction was that if **I** were an IT guy and it was possible and practical to operate behind what is essentially a **whitelisting Internet firewall**, if that only caused a tiny bit of trouble in fringe cases, that would be entirely acceptable for my peace of mind and in return for the enhanced security it would provide for everyone else all of the time. And what's cool is that it protects against the unknown. The widespread Solarwinds vulnerability which took the whole world by surprise would not have affected any of their users and the always evolving Pegasys smartphone spyware would not function if it were to infect any of the handsets protected by this system.

As usual, my focus and interest was on the technology, which I think is clearly pretty cool and which I felt an obligation to share. Although I cannot vouch for this from first hand experience, I do know these guys and I've known them for nearly 20 years. They're techies like us. Just take a look at any of the many videos posted on their website and at how excited they are about what they have created and you'll know that immediately. I was **very** impressed by what I saw last Friday. I don't know anything about the business side of this. My sense is that the solution is scalable from a single router in a small office to a global enterprise – but I don't know that for sure. I also have no idea what it costs. You'll need to track that down for yourselves if this sounds interesting. So, anyway... after what I heard and learned Friday, I wanted to put this on everyone's radar. They are AdamNetworks at https://adamnet.works and I have some links in today's show notes for their site though I just got them by poking around a bit yesterday. You'll find a page of videos and even their wide open tech support portal:

- https://adamnet.works/
- https://vimeo.com/adamnetworks/videos
- https://support.adamnet.works/latest
- https://support.adamnet.works/t/adam-one-operational-instructions-and-guidelines/70
- https://docs.google.com/document/d/e/2PACX-1vTJ22Jlxc3mxhvvrsPcdG_D0qAg0SCu1cfxOtuERqOCQs4KgYlj0fMuZLQnx9aL4cZNAHVW1oGIUOGx/pub#h.j50hqa26l5i

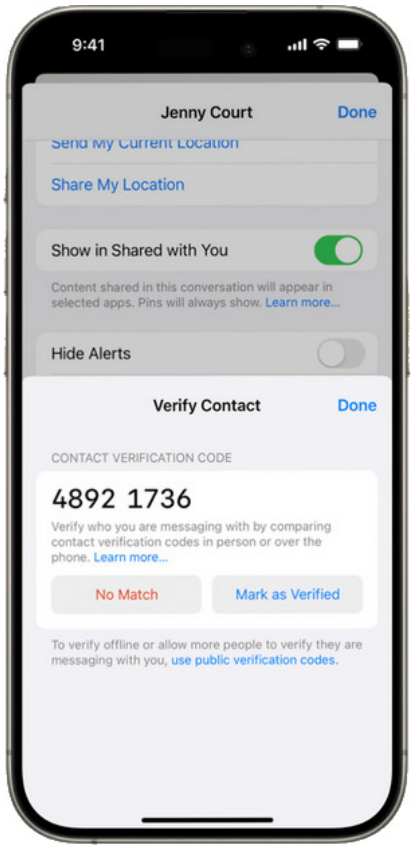
Okay... on with what I think is going to be an interesting podcast!

Security News

An update on stolen LastPass vault decryption:

Just last week, the still unknown hackers who breached LastPass and exfiltrated everyone's encrypted vaults managed to siphon off another $4.4 million worth of crypto-assets from 25 of LastPass' previous or current customers. We know that most LastPass users will have remained with LastPass and won't be receiving the news that stolen vaults are being actively decrypted. As we saw previously, the hackers are cracking the stolen LastPass password vaults, presumably those that were protected by weak passwords and few PBKDF iterations. From those vaults they are recovering crypto-wallet seed phrases, and draining those customers' user accounts. So, with the latest round of thefts, this brings the total to nearly $40 million after they previously stole another $35 million from ~150 other users earlier this year.

iMessage gets Contact Key Verification



Remember that cool feature of Threema that I always liked so much? It was the ability to positively verify the other person's public key through an out-of-band communication channel, like a phone call or, even better, a face-to-face meeting. Apple hasn't quite gone that far, since they continue to manage all of their user's keys. But they have developed a slick way to verify any new device that registers itself with a user's account. They call it "iMessage Contact Key Verification" and it has appeared in the developer previews of iOS 17.2, macOS 14.2 and watchOS 10.2. It's designed to present an alert inside iMessage conversations whenever a new device is added to a participant's account.

Say that I'm in a message dialog with someone and they add a new device to their account. How do I know that the new device was added by the person I think I'm messaging with? If it was someone else holding the new device that's not good. So this new system displays a fixed 8-digit code in both your phone and that newly added phone. This allows you to arrange a phone conversation or a Facetime chat to compare verification codes, which should match. Apple's on-screen instructions say: *"Verify who you are messaging with by comparing contact verification codes in person or over the phone."* The idea is that by seeing the other person who has just added a new device to their account, you agree that you're both seeing the same code, which adds that device into the connection, verifying that it's owned by who you think it is.

Perhaps there's hope for longer Win10 support?

It turns out that Windows 10 is good for the Earth... or more correctly it's that Windows 11 is less so. More than 20,000 members of the Public Interest Research Group (PIRG) have formally asked Microsoft to extend support for Win10, which, as I recently mourned, is scheduled to reach End-of-Life status in two years, in October 2025.

The organization says that, get this, **around 40% of the one billion devices that run Windows 10** cannot be upgraded to Windows 11, and will create an unnecessary e-waste problem. Of course, on the flip side, Microsoft and many other hardware vendors are looking forward to this event for just that reason. PIRG is the same industry group that had recently convinced Google to extend the warranty and support for older Chromebooks using similar arguments. It's certainly somewhat bracing to learn that 40% of current machines won't be compatible with Windows 11. That's a legitimately huge deal. I wonder whether Microsoft is prepared for the user – and corporate – ire that will follow from that. Interesting times!

Bug bounty news from HackerOne

Last Thursday, the HackerOne bug bounty folks took the occasion of their total bounty payouts crossing the $300 million dollar mark to provide some insights into the current state of their operations, which I've edited a bit. Just to be clear in the following text, when they refer to "customers" they mean organizations that pay bounties to have their bugs found. And, of course, "hackers" are the freelance agents who find those bugs. So, they wrote:

HackerOne today announced its ethical hacker community has surpassed $300 million in total all-time rewards on the HackerOne platform. And 30 hackers have also earned more than one million dollars on the platform, with one hacker surpassing $4 million dollars in total earnings. And hackers are finding new opportunities to earn more by diversifying their skill sets as emerging technology reshapes the threat landscape. 55% of hackers surveyed expect that Generative AI will become a top target in the future. Crypto and blockchain organizations continue to see strong program engagement — offering the highest average overall rewards for hackers, including the award of this year's top payout of $100,050. HackerOne's customers have also expanded how they commission hackers outside of traditional bug bounties with pentesting engagements increased by 54% on the platform in 2023.

- *Hackers continue to experiment with GenAI, as 61% of hackers said they will use and develop hacking tools from GenAI to find more vulnerabilities, and another 62% of hackers plan to specialize in the OWASP Top 10 for Large Language Models. Hackers also said they plan to use GenAI to write better reports (66%) or code (53%) and reduce language barriers (33%).*

- *Hackers report insufficient in-house talent and expertise as the top challenge for organizations, which is a gap they are filling: 70% of customers stated that hacker efforts have helped them avoid a significant cyber incident.*

- *Fifty-seven percent of HackerOne customers believe exploited vulnerabilities are the greatest threat to their organizations, over phishing (22%), insider threats (12%), and nation-state actors (10%).*

- *Customers are getting faster at fixing vulnerabilities, as the average platform-wide remediation time dropped by 10 days in 2023. (I would like to know "dropped from what to what"? Did it drop from 20 days to 10 days or from 100 days to 90 days? But their summary didn't say.) Interestingly, automotive, media/entertainment, and government verticals saw the biggest decrease in time to remediation with an over 50% improvement.*

- *And organizations are reducing costs by embracing human-centered security testing earlier in their software development lifecycles, with customers saving an estimated $18,000 from*

security experts reviewing their code before release.

Chris Evans, HackerOne's CISO and Chief Hacking Officer was quoted, saying: "Organizations are under pressure to adopt GenAI to stay ahead of competitors, which, in turn, is transforming the threat landscape. If you want to remain proactive about new threats, you need to learn from the experts in the trenches: hackers.

That annual Hacker-Powered Security Report is based on data from HackerOne's vulnerability database and gathered the views from both HackerOne customers and more than 2,000 hackers on the platform. It seems that an interesting takeaway is that mistakes are bound to be made in the rush to get new Generative AI solutions rolled out. So getting up to speed on AI technology appears to be on everyone's radar.

Logging Made Easy (from your friends at CISA!)

Here's an interesting item: Last Friday, our own CISA released a tool called Logging Made Easy through CISA's official GitHub account. LME is a toolkit that enhances log management on Windows devices. The tool was originally developed by the UK's NCSC agency way back in the late 2010's and they retired it in March of this year. So, CISA picked up the tool, updated and re-wrote it to cover recent Windows versions and logging capabilities. Here's what CISA said:

Today, CISA announces the launch of a new version of Logging Made Easy (LME), a straightforward log management solution for Windows-based devices that can be downloaded and self-installed for free. CISA's version reimagines technology developed by the United Kingdom's National Cyber Security Centre (NCSC), making it available to a wider audience.

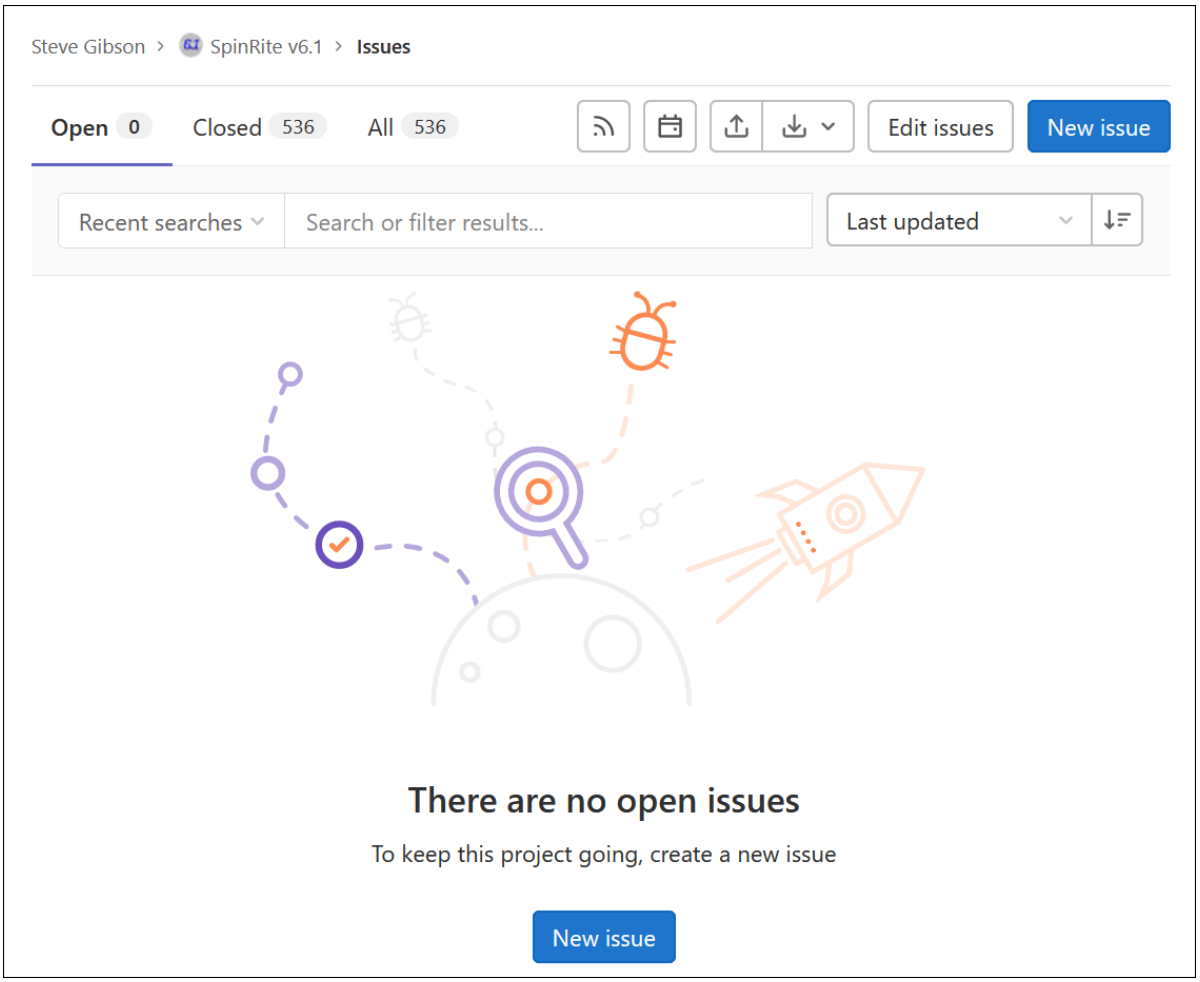
Log management makes systems more secure. Until now, it has been a heavy lift for many targeted organizations, especially those with limited resources. CISA's LME is a turnkey solution for public and private organizations seeking to strengthen their cybersecurity while reducing their log management burden. LME builds upon the success of the NCSC's log management solution and CISA urges organizations to secure their Windows-based devices today by downloading the free LME technical solution.

Over on their GitHub page they explain that Logging Made Easy can:

- Show where administrative commands are being run on enrolled devices
- See who is using which machine
- In conjunction with threat reports, it is possible to query for the presence of an attacker in the form of Tactics, Techniques and Procedures (TTPs)

Since that all seems pretty cool and useful, the link to jump there is this week's GRC shortcut: https://grc.sc/946 But the full link is also not difficult, it's: https://github.com/cisagov/LME

SpinRite:



On Sunday, SpinRite's second and quite possibly final pre-release was published. As far as I know at this point, the DOS executable code that's now available will be what finally ships. SpinRite 6.1 includes a tiny DOS text editor for viewing and editing its log files from DOS, and to make editing any DOS config files easier. That editor also appears to be ready. I need to verify that all of the FreeDOS OS kernel customizations I've made for SpinRite are complete, then I'll return to finish a bit of work on the Windows side of SpinRite, incorporating improvements that arose from the work on ValiDrive. I need to update GRC's servers to digitally sign individually licensed copies of SpinRite on-the-fly for download. At that point, SpinRite 6.1 will be available as a finished product to all SpinRite 6 owners and I'll begin work on SpinRite's web pages to document its operation with all of the many changes and improvements that have been made during the past three years of this project.

And after I get an eMail system in place to begin notifying SpinRite 6.0's 19 years worth of existing users, I get to start on SpinRite 7 ... which really excites me! But in the meantime, we really do have a major step forward. This morning I saw this posting in GRC's web forums:

> *Thanks to the Pre-release of 6.1 I'm running SpinRite on drives that I either haven't been able to run it on for years, or drives that were too big for 6.0, so I was never able to run SpinRite on them before.*

Closing the Loop

Shep Poor / @sheppoor

Hi Steve, you've trained us well. When I saw a bleeping computer article titled "Windows 11 adds support for 11 file archive formats" my first thought was "oh great. 11 new interpreters built into Windows, ready to be exploited".

I appreciated Shep's Tweet because "thinking security" is probably the best thing our listeners could take away from the many lessons we learn here every week from specific events. Those events will come and go. But seeing the underlying issues will endure and will remain valuable long after the UNIX epoch.

B P / @WildlandsGuy

Hi Mr. Gibson. I had a huge smile on my face when you announced you'll be doing SN past 999. I would appreciate hearing your opinion on what it would take to get the world to move to IPv6. What are the significant hurdles? (FYI - Ran Validrive on 3x PNY drives from BestBuy and all checked out ok. :)

There are probably still some backwaters of the Internet which IPv6 hasn't reached. But from everything I'm seeing, nearly everything appears to be IPv6 ready. So I think that at this point the only thing that's holding things back is a concern over breakage – what might break if IPv4 was withdrawn and we were left only with IPv6? We've seen examples where surprising things have broken by change. We were just talking about the surprise that the designers of the upgraded TLS v1.3 protocol faced when it initially didn't work because too many boxes somewhere in the traffic path broke. They shouldn't have broken, but they did. So it was necessary for the TLS v1.3 guys to make 1.3 appear to be much more like 1.2.

And we've seen how incredibly cautious the browser designers are when they change anything. They'll initially have some new fangled feature disabled and only enabled manually by intrepid developers. Then they'll turn it on by default for 1% of the population while closely monitoring it for trouble, then they'll continue to gradually roll it out, enabling it more and more while still being super cautious.

So, in the case of IPv6, I think we're at the "if it's not broke, don't fix it" stage. The good news is that everyone is probably ready for it when it's forced upon us. But until it's truly necessary form someone to not have IPv4, until there just isn't any IPv4 left and the only thing that newcomers can have is an IPv6 address, that's when I suspect that IPv6 will finally begin in earnest – and really no sooner – not until there's really no other choice.

Not / @NotDorion

Hi Steve, I've recently seen the high cost of signing certificates, especially for open-source projects like ImageMagick: https://github.com/ImageMagick/ImageMagick/discussions/6826

On one hand, I see that having costly certificates makes it difficult for malware to access

them. But maybe a solution is needed for open-source projects? Would be glad to hear your thoughts on this! Thanks for all the great content all these years!

So this was interesting. The posting on GitHub that our listener linked to was titled *"The Windows installer of ImageMagick will no longer be signed"* and its developer wrote:

"Today our code signing certificate will expire. For many years LeaderSSL sponsored us with a code signing certificate but they are no longer able to do so. Since June of 2023 the CA/B Forum requires that OV code signing private keys be stored on a FIPS 140-2 Level 2 or Common Criteria Level EAL4+ certified device. This means we are no longer able to export our code signing certificate with its private key and use this in GitHub actions. We would now either need to have our own GitHub agent and hardware token or use a cloud solution (e.g. digicert). Our preference would be to use a cloud solution that integrates with GitHub. Digicert seems to be our only option now but a certificate there would cost $629 (tax excluded) for a single year. If your organization requires a signed installer then please consider sponsoring us with a code signing certificate. Please reach out to @dlemstra for questions or in case of a sponsorship."

What was neat was that in a terrific example of community action, this posting from three days ago generated a terrific thread which, based upon this developer's reaction, appeared to provide a number of useful alternative solutions. One being a free solution for a year and others involving Azure Code Signing. I didn't dig into it any further, but other open source projects are having similar difficulties and they chimed into the thread. So I wanted to share the news of this in the event that the eventual solution found by ImageMagick's developer might be use to some of our listeners. As we know, all of GRC's code is signed by a DigiCert EV certificate which works for me since I'm a commercial entity,

Matt Davis / @itsthemattdavis

Offered some great feedback commentary about last week's look info privilege:

Hi Steve, thrilled to hear you are continuing past episode 999 - I've listened since the beginning and I'm looking forward to another 999 or so afterwards.

You may have overlooked the heart of the Privileged Accounts issue you discussed during last week's episode. The problem isn't lazy IT folks signing in as "root" in the morning, it's that we are trying to create a shift away from 1-to-1 parity between Accounts and People.

Many moons ago, shared logins were the norm for system management, which has obvious downsides and drawbacks we all understand. Single Sign-On solutions like OKTA aimed to solve this by promoting unique accounts per user. Each person gets exactly enough privilege for their needed tasks, and staff turnover is easy to handle by revoking a single password. However, this led to a new problem: administrators only have a single account, which they have no choice but to use, even for basic tasks - and it's all powerful.

The obvious solution would be to create separate Standard & Admin accounts for every administrator, but since most software now is charged per-account-per-month, this tends to be cost prohibitive (we have one service at $300/month/user...). Ideas like Just Enough

Access (JEA) help, allowing admins to switch between regular and elevated access, akin to UAC in specific environments like Azure and AWS. But slow adoption in userland means that I still need to use my SuperUser-level login to access my email every morning.

Looking forward to having the option to submit via e-mail, it seems like the Feedback Form hasn't really been working over the last few years and you are the only reason I've pulled my twitter account out of the dustbin. Love the show and SpinRite, thank you!

Some terrific perspective from the field. Thanks, Matt! Charging an organization $300 per user per month for authentication makes my blood boil. What a ripoff!

Jpmcneal / @jpmcneal

Hi, Steve. I'm huge fan of SN and so glad you'll keep going after episode 999. I'm wondering what your thoughts are on turning off TLS 1.2? I did a little research and found it interesting that there's no official EOL for TLS versions. It's just a matter of when the big browser vendors decide to not support it. (Microsoft, Apple & Google agreed to stop supporting TLS 1.0 in 2018 for example). Thank you, JP PS: I work at a hosting company.

I suppose time flies... but my impression is that TLS 1.2 is still a relatively recent arrival and that 1.3 hasn't fully happened yet. TLS 1.0 and 1.1 were both deprecated only two years ago in 2021. TLS 1.2 has been in use since 2008 and TLS 1.3 only began ten years later and five years ago in 2018. But here's the stat that most matters: As of last month, 99.9% of all websites support TLS 1.2 but only **64.8% support TLS 1.3.** So not even 2/3rds of the Internet's websites are TLS 1.3 capable today. And, given a good choice of cipher suites for TLS v1.2, there's really nothing wrong with it. So the Internet as a whole is not going to be able to move beyond TLS 1.2 for a while yet... not until adoption of 1.3 becomes far stronger than it currently is. And that'll just take time. New web servers will all be supporting 1.3 but older servers which only support 1.2 will need to either be upgraded or die and be replaced with new servers.

CitrixBleed

I decided to punt our continuing exploration of the joint NSA/CISA Top 10 CyberSecurity Misconfigurations until at least next week due to another mass casualty cybersecurity event that gives us a really interesting opportunity to take our podcast listeners on a journey into the interior code guts of a widely used and hugely popular piece of networking hardware whose recent wide-scale compromise is believed to have negatively impacted the network security of more than 20,000 enterprises, worldwide, who had been depending upon the integrity of this device to keep them protected. So this week we're going to take a deep dive into the specific code flaw that resulted in the coining of the term **CitrixBleed**.

I've often spoken of the now-common practice of jumping on, and quickly reverse engineering, vulnerability-closing update patches as a means of discovering the details of the defect that the patch corrects for the purpose of either (a) if you're a white hat security research firm, documenting an interesting piece of Internet history or (b) if you're a black hat criminal organization, using the information gleaned from such reverse engineering to begin compromising the hapless users of not-yet-patched devices.

As always, it's one thing to talk about this being done and an entirely different thing to obtain a good sense of the details from someone who is actually doing it. So I smiled when I encountered a write-up from a firm wearing a white hat of the process they underwent to reverse engineer a really really bad vulnerability that is, unfortunately, being exploited and is damaging people right now. Before I get into that nitty-gritty, I want to set the stage by sharing an overview of the situation written by security researcher Kevin Beaumont, whom we often encounter in the security space, tweeting as "Gossi The Dog". Here's how Kevin described the situation three days ago. Under his headline *"Mass exploitation of CitrixBleed vulnerability, including a ransomware group"* he wrote:

> *Three days ago, AssetNote posted an excellent write up about CitrixBleed aka CVE-2023–4966 in Citrix Netscaler. This vulnerability is now under mass exploitation. A few weeks ago it was under limited targeted exploitation to allow network access. It's not AssetNote's fault — it was clear multiple groups had already obtained technical details.*
>
> *The patch became available on October 10th [— so exactly 3 weeks ago, today —]. Even if you applied the patch and rebooted, you still have a problem as session tokens persist. The vulnerability allows memory access. Sounds boring, right? That same memory contains session tokens, which an attacker can easily extract.*
>
> *Those session tokens allow the bypass of needing login credentials and the bypass of all multi-factor authentication — an attacker can just replay the session key, and they're in.*
>
> *Who uses Citrix Netscaler anyway?! Many tens of thousands of businesses run it. It is very, very common in enterprises and governments. If you think nobody runs this stuff, you probably also think everybody uses Linux on their laptop.*

For an additional perspective, the Risky Business Newsletter wrote:

A Citrix vulnerability has entered the dangerous stage of mass exploitation as multiple threat actors are compromising unpatched devices all over the internet in a race with each other to steal their session tokens.

Known as CitrixBleed and tracked as CVE-2023-4966, the vulnerability impacts Citrix ADC and Citrix NetScaler, which are extremely complex networking devices used in large enterprise and government networks in multiple roles, such as gateways, proxies, caching, VPN servers, and a bunch of other stuff.

The vulnerability allows threat actors to send junk data to the Citrix OpenID component that will crash and leak a part of the device's memory. The bad part is that, in some cases, this memory may contain session tokens that attackers can collect and then bypass authentication to access the device [and the network behind.]

Citrix released patches to fix the CitrixBleed memory leak earlier this month, on October 10.

The bug in itself is extremely bad as it is, but things took a turn for the worse a week later when Google Mandiant researchers came out to say they found evidence CitrixBleed had been exploited in the wild since late August—making the vulnerability one of this year's tens of actively exploited zero-days.

Mandiant said a threat actor was using the bug to gain access to Citrix gateways and then pivot to internal systems. The attacks were small in scale and targeted professional services, technology, and government organizations.

Things went from bad to disastrous last week, around October 23-25, when several proof-of-concept exploits started popping up on GitHub and vulnerability portals. Within a day, mass-exploitation was in full swing. At the time of writing, GreyNoise is tracking more than 120 IP addresses that are probing the internet and attempting to exploit CitrixBleed.

Kevin Beaumont later posted an update over on Mastodon:

Quick update on #CitrixBleed - tracking just over 20,000 exploited Netscaler servers so far today, where session tokens have been stolen. How? Have a honeypot running to gather data on attackers, then compare with Netflow via industry friends. Two TCP connections (first large) plus Shodan cross reference to validate Netscaler victim. Also it turns out in March of this year somebody documented how to replay the session token to bypass MFA/login. https://www.vulnerability-db.com/?q=articles/2023/07/03/citrix-gateway-cloud-mfa-insufficient-session-validation-vulnerability

Okay. So on the back of last week's Cisco web portal disaster, we now have another web portal login authentication disaster from another manufacturer, Citrix, who has a similar long history of catastrophic security vulnerabilities. And Kevin has arranged to confirm more than 20,000 instances of actual login session token exfiltration.

So now let's look at the mixed blessing arising from the need to patch a horrible and widespread problem in full public view. What steps do both curious security researchers and rapacious malicious criminals take to turn a patch into an exploit? The security research firm "AssetNote" explains. They wrote:

It's time for another round of Citrix Patch Diffing! Earlier this month Citrix released a security bulletin which mentioned "unauthenticated buffer-related vulnerabilities" and two CVEs. These issues affected Citrix NetScaler ADC and NetScaler Gateway.

We were interested in CVE-2023-4966, which was described as "sensitive information disclosure" and had a CVSS score of 9.4. The high score for an information disclosure vulnerability and the mention of "buffer-related vulnerabilities" piqued our interest. Our goal was to understand the vulnerability and develop a check for our Attack Surface Management platform.

For those unfamiliar with Citrix NetScaler, it is a network device providing load balancing, firewall and VPN services. NetScaler Gateway usually refers to the VPN and authentication components, whereas ADC refers to the load balancing and traffic management features. We have covered issues in NetScaler several times before.

We began by installing and configuring the two versions we wanted to compare. We chose 13.1-49.15 and 13.1-48.47. From our previous work with NetScaler we knew to look in the /netscaler/nsppe binary. This is the NetScaler Packet Processing Engine and it contains a full TCP/IP network stack as well as multiple HTTP services. If there is a vulnerability in NetScaler, this is where we look first.

We decompiled the two versions of nsppe with Ghidra and used the BinExport extension to create a BinDiff file.

Four years ago we discussed Ghidra at some length. It's a free and open source binary code reverse engineering tool which was initially developed by the US National Security Agency, our NSA. The NSA initially released the Ghidra binaries during the 2019 RSA Conference and then followed-up by releasing its source code a month later on GitHub. Before Ghidra, the proprietary and quite expensive IDA Pro was the only real game in town. Now that's no longer the case.

Resuming what AssetNote was explaining, they had said: *"We decompiled the two versions of nsppe with Ghidra and used the BinExport extension to create a BinDiff file."* You can probably guess, but "BinDiff" is short for "Binary Difference" – and this, of course, is the key to the attack. When something is patched, some code somewhere is changed. So the first challenge is to locate the regions that the patch has changed, then to carefully examine the before and after code to understand what problem that change was intended to fix. They wrote:

This process takes a while as the binaries are quite large. To ensure success we tweaked the decompiler settings under Edit -> Tool Options -> Decompiler to the following.

Cache Size (Functions): 2048
Decompiler Max-Payload (Mbytes): 512
Decompiler Timeout (seconds): 900
Max Instructions per Function: 3000000

After creating the BinDiff files we opened them up for comparison and found roughly 50 functions had changed. We proceeded to check each one, often opening both versions in Ghidra and comparing the decompiled output with a text diffing tool.

*We found two functions that stood out **ns_aaa_oauth_send_openid_config** and **ns_aaa_oauthrp_send_openid_config**. Both functions perform a similar operation, they implement the OpenID Connect Discovery endpoint. The functions are both accessible unauthenticated via the **/oauth/idp/.well-known/openid-configuration** and **/oauth/rp/.well-known/openid-configuration** endpoints respectively.*

*Both functions also included the same patch, an additional bounds check before sending the response. This can be seen in the snippets below showing the before and after for **ns_aaa_oauth_send_openid_config**.*

The original code:

```
iVar3 = snprintf(print_temp_rule,0x20000,
                 "{\"issuer\": \"https://%.*s\", \"authorization_endpoint\": \
                 ,uVar5,pbVar8,uVar5,pbVar8,uVar5,pbVar8,uVar5,pbVar8,uVar5,pb
authv2_json_resp = 1;
iVar3 = ns_vpn_send_response(param_1,0x100040,print_temp_rule,iVar3);
```

And the updated code:

```
uVar7 = snprintf(print_temp_rule,0x20000,
                 "{\"issuer\": \"https://%.*s\", \"authorization_endpoint\": \
                 ,uVar5,pbVar8,uVar5,pbVar8,uVar5,pbVar8,uVar5,pbVar8,uVar5,pb
uVar4 = 0x20;
if (uVar7 < 0x20000) {
    authv2_json_resp = 1;
    iVar3 = ns_vpn_send_response(param_1,0x100040,print_temp_rule,uVar7);
    ...
}
```

I've grabbed the two code snippets for anyone who's interested in reading the code since the change is so clear. But I can describe what's going on for those who are listening: In the original code, the amount of data that is returned in response to the query is determined by the size of a string that results after substituting a whole bunch of variables into a string template. After all of those substitutions are made, the result will be a much larger string, so a 128K character buffer is provided to contain this expanded substituted string.

The string substitution function, known as "snprintf" will be quite familiar to C programmers. "Snprintf" performs the string substitution up to the limit of the length of the buffer that it's been given, then it returns as its function return value the full length that the string would be after performing all of the substitutions. It will **not** overflow the size of the buffer it's been provided. But it **will** return a larger value than the size of the buffer it's been provided if the resulting substituted string will not fit into the provided buffer. This is typically done to tell the programmer how large a buffer would be needed to hold the entire expanded string if the buffer that was provided was not large enough. Code that cared about that would then typically allocate a new buffer that was large enough, then perform the string substitution again into the large-enough buffer. But in any event, this is a safe string function that will not overflow the size of the buffer it has been provided.

So here comes the fatal flaw: In the original code, immediately after "snprintf" does its work and returns the size of the **required** buffer, that size is directly used to specify the length of the data that will be returned from the user's (or attacker's) query. In other words, what the original coder who wrote this **failed to do** was to check that the size of the response was not larger than the buffer which had been set aside to contain it.

Failing to perform that simple and required check meant that if the remote attacker could somehow arrange to cause that "snprintf" function to assemble a huge long string, and thus specify a massive response size, even though "snprintf" would not itself overflow the much smaller 128K buffer it had been provided, the response that would be sent back would be massive, with the data in that response coming from whatever happened to be lying around in RAM after that 128K buffer.

And, as it turns out, that just as with the infamous HeartBleed vulnerability where the web server's private key just might have happened to be in nearby RAM, in the case of this CitrixBleed disaster, the RAM following that 128K output buffer is chock full of valid logged-on authentication tokens which the server dutifully sends back in response to the attacker's query... thus allowing any otherwise unauthenticated remote attacker to become authenticated by those tokens and thus logged on.

What AssetNote discovered in the patched and updated code, the change in the code that drew them to that spot, was the addition of a simple check before sending **any** reply, to verify that the value returned by the preceding "snprintf" function was less than 128K. In other words, the result of formatting the reply to the user would fit within the 128K buffer that was provided. And if not, the bogus query would be ignored and no response would be returned to the user – or any attacker.

And as for exploiting this, AssetNote explained:

> *To exploit this, all we needed to do was figure out how to get the response to exceed the buffer size of 128K bytes. The application would then respond with the completely filled buffer, plus whatever memory immediately followed the print_temp_rule buffer.*
>
> *Initially we thought the endpoint would probably not be exploitable. The only data that was inserted was the hostname, which is something that needed administrator access to configure.*

Luckily for us, we were wrong and the value inserted into the payload did not come from the configured hostname. It actually came from the HTTP Host header.

Which is, as we know, and as we were just talking about recently, one of the several query headers in any HTTP query. So it is entirely within the remote attacker's control. In other words, "whoopsie!" They wrote:

*We were also fortunate that NetScaler inserts the hostname into the payload **six times**, as this meant we could hit the buffer limit of 128K bytes without running into issues because either the Host header or the whole request was too long.*

In other words, there was also a coincidental buffer size amplifier that worked to make attacks based upon this mistake practical.

We might assume that the coder of the original implementation knew that the "snprintf" function was string safe, in that it could not be induced to overflow the buffer that was provided. But rather than using the actual size of the string that ended up being returned in the buffer, the coder made the mistake of using the value returned by the "snprintf" function, which is, as we all now know, not the same as the length of the string that it placed into the buffer. If the string that it **wanted** to place into the limited-size buffer was larger than the buffer, it would return that size. And, unfortunately, the programmer used **that** value as the length of the reply to send. Echoing these thoughts, AssetNote concluded:

*Here we saw an interesting example of a vulnerability caused by not fully understanding **snprintf**. Even though **snprintf** is recommended as the secure version of **sprintf** it is still important to be careful. A buffer overflow was avoided by using **snprintf** but the subsequent buffer over-read was still an issue.*

Like previous issues with Citrix NetScaler, the issue was made worse by a lack of other defense in depth techniques and mitigations. Not clearing sensitive data from what appear to be temporary buffers and stricter validation on client provided data being the two most obvious mitigations which could have been applied to minimize the damage.

So, as a consequence of that small oversight which was apparently first discovered back in August – but also enabled by, as AssetNote observed – a more general lack of architectural care in a security appliance, the network integrity of more than 20,000 large and high-end enterprise and government users of these network appliances has been compromised. And not just found vulnerable but actually known to have been exploited. It's possible to do better. When I was working on SQRL's code, I had a window up on the screen which continually and in real time showed me the current RAM contents of all sensitive data. I had deliberately grouped it all into a single region for easier monitoring and wiping and anything sensitive was wiped from RAM the moment it was no longer needed and I was always able to confirm that visually.