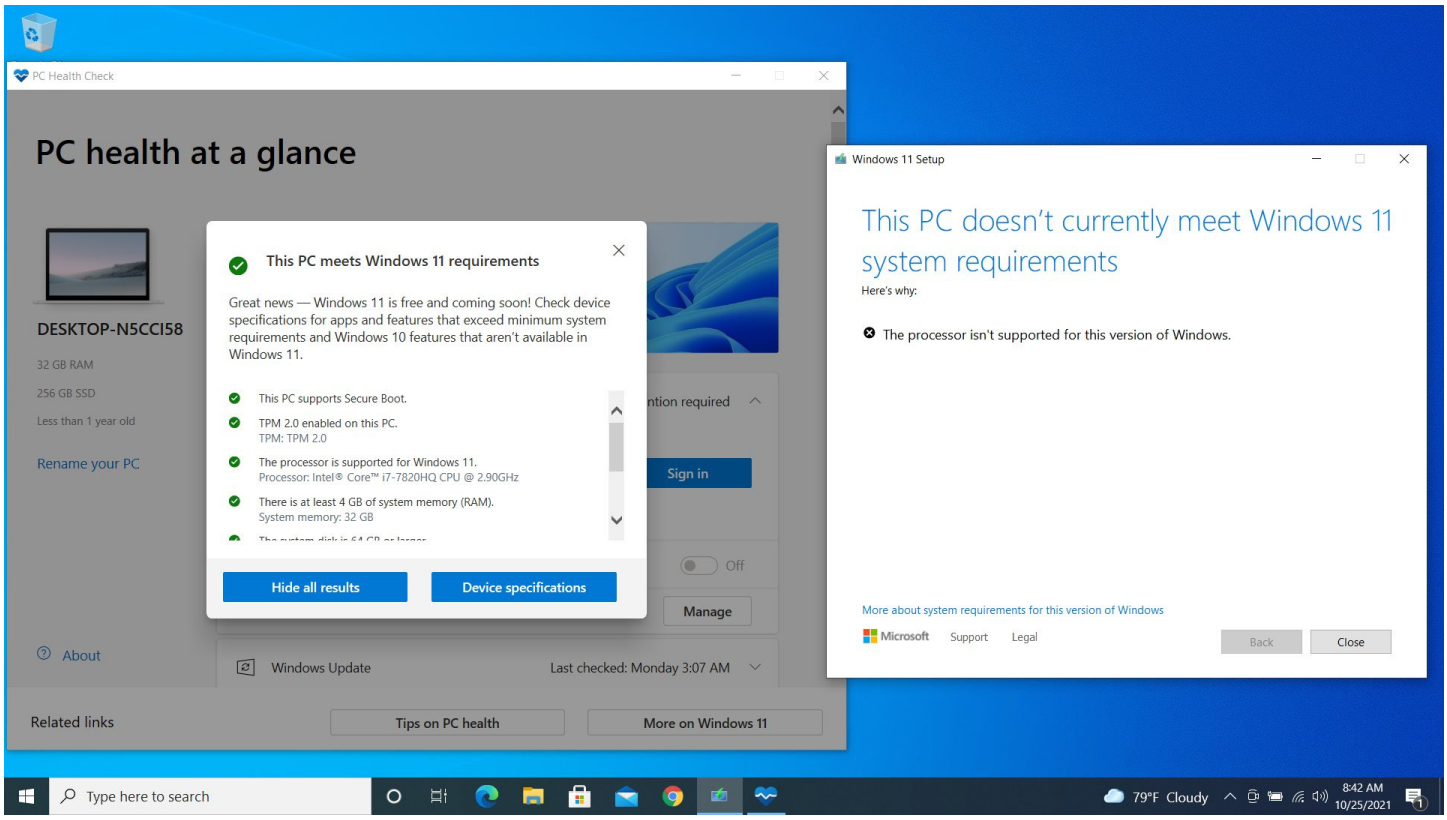
Security Now! #843 - 11-02-21

“Trojan Source”

This week on Security Now!

This week we keep counting them Chrome 0-days, we look at a pair of badly misbehaving Firefox add-ons with Mozilla's moves to deal with their and future proxy API abuse. We check-in for Windows news from Redmond which I'm again unable to resist commenting upon, then we look at a surprise motherload of critical updates from Adobe and at the still-ongoing DDoS attacks against VoIP providers and their providers. We'll look at some fun and interesting Closing The Loop feedback from our listeners and I'm able to share some surprising early benchmarks from SpinRite. Then we finish by looking at a frighteningly clever and haunting new attack against source code known as “Trojan Source”.

Quality Workmanship



StarKiss / @StarKissedOne
Apparently my Dell Core-I7 is both compatible and incompatible with Win11... at the same time. Best quote from last show, "Win11 for people who like pain." Love the show!

0-Day Watch

More 0-days for Chrome

Two newly discovered critical true in-the-wild 0-day's were patched in Chrome's most recent urgent release last Thursday when Google pushed out an emergency update, bringing Chrome to v95.0.4638.69.

Tracked as CVE-2021-38000 and CVE-2021-38003, the problems were an insufficient validation of untrusted input in a feature called "Intents" as well as a case of "inappropriate implementation" (whatever the heck that means!) in V8, Chromium's JavaScript & WebAssembly engine. Both discoveries are credited to TAG, Google's Threat Analysis Group, which reported their findings to the Chromium team on September 15, 2021 and October 26, 2021 respectively.

Google's sparse advisory regarding these simply stated that: "Google is aware that exploits for CVE-2021-38000 and CVE-2021-38003 exist in the wild." But the fact that an actively exploited in-the-wild 0-day was reported to the Chromium team on September 15th and only patched last Thursday does seem a bit slow for them, since they typically fix problems within a few days. Presumably in this instance they had their reasons.

In addition to these two biggies, the other vulnerability of note that was repaired was the use-after-free vulnerability in Chrome's Web Transport component (CVE-2021-38002), which was leveraged and demonstrated for the first time at the Tianfu Cup contest held the middle of last month in China.

And... since we've been counting, these two latest emergency patches bring Chrome's resolved while being actively in-use 0-day updates to a record 16 0-days since the start of the year...

1. CVE-2021-21148 – Chrome 88.0.4324.150, on February 4, 2021.
2. CVE-2021-21166 – Chrome 89.0.4389.72, on March 2, 2021.
3. CVE-2021-21193 – Chrome 89.0.4389.90, on March 12, 2021.
4. CVE-2021-21206 – Chrome 89.0.4389.128, on April 13, 2021.
5. CVE-2021-21220 – Chrome 89.0.4389.128, on April 13, 2021.
6. CVE-2021-21224 – Chrome 90.0.4430.85, on April 20, 2021.
7. CVE-2021-30551 – Chrome 91.0.4472.101, on June 9, 2021.
8. CVE-2021-30554 – Chrome 91.0.4472.114, on June 17, 2021.
9. CVE-2021-30563 – Chrome 91.0.4472.164, on July 15, 2021.
10. CVE-2021-30632 – Chrome 93.0.4577.82, on September 13, 2021.
11. CVE-2021-30633 – Chrome 93.0.4577.82, on September 13, 2021.
12. CVE-2021-37973 – Chrome 94.0.4606.61, on September 24, 2021.
13. CVE-2021-37975 – Chrome 94.0.4606.71, on September 30, 2021.
14. CVE-2021-37976 – Chrome 94.0.4606.71, on September 30, 2021.
15. CVE-2021-38000 – Chrome 95.0.4638.69, on September 15, 2021.
16. CVE-2021-38003 – Chrome 95.0.4638.69, on October 26, 2021.

This is the greatest number of 0-days Google has ever patched in Chrome during any calendar year since Chrome's debut 13 years ago, in 2008.

Browser News

Two naughty Firefox add-ons have been caught abusing an extension API.
Mozilla's Security Blog posting was titled: "Securing the proxy API for Firefox add-ons"

https://blog.mozilla.org/security/2021/10/25/securing-the-proxy-api-for-firefox-add-ons/

Mozilla sets things up by explaining: *"Add-ons are a powerful way to extend and customize Firefox. At Mozilla, we are committed not only to supporting WebExtensions APIs, but also ensuring the safety and reliability of the ecosystem for the long term."* Then they explain what happened...

In early June, we discovered add-ons that were misusing the proxy API, which is used by add-ons to control how Firefox connects to the internet. These add-ons interfered with Firefox in a way that prevented users who had installed them from downloading updates, accessing updated blocklists, and updating remotely configured content. In total these add-ons were installed by 455,000 users.

This post outlines the steps we have taken to mitigate this issue as well as provide details of what users should do to check whether they are affected. Developers of add-ons that use the proxy API will find some specific instructions below that are required for future submissions.

So, the two malicious add-ons Mozilla found were blocked, to prevent their installation by other users. Their names are "Bypass" and "Bypass XM". They wrote:

To prevent additional users from being impacted by new add-on submissions misusing the proxy API, we paused on approvals for add-ons that used the proxy API until fixes were available for all users.

Starting with Firefox 91.1 (we're now at 93), Firefox includes changes to fall back to direct connections when Firefox makes an important request (such as those for updates) via a proxy configuration that fails. Ensuring these requests are completed successfully helps us deliver the latest important updates and protections to our users. We also deployed a system add-on named "Proxy Failover" (ID: proxy-failover@mozilla.com) with additional mitigations that has been shipped to both current and older Firefox versions.

So, in other words, if Firefox's own checking for and downloading updates fails, it will now feel free to bypass the bypass to go after its own updates directly.

Under "As a Firefox user, what should I do next?" they wrote: *"It is always a good idea to keep Firefox up to date, and if you're using Windows to make sure Microsoft Defender is running. Together, Firefox 93 and Defender will make sure you're protected from this issue."*

That's interesting about Windows Defender. Windows Defender will catch and block this behavior, and in other reporting on this I saw them note that Windows Defender is currently the only A/V system that protects Firefox users from these threats.

So, Mozilla says:

> First, check what version of Firefox you are running. Assuming you have not disabled updates specifically, you should be running at minimum the latest release version, which is Firefox 93 as of today (or Firefox ESR 91.2). If you are not running the latest version, and have not disabled updates, you might want to check if you are affected by this issue. First, try updating Firefox. Recent versions of Firefox come with an updated blocklist that automatically disables the malicious add-ons. If that doesn't work, there are a few ways to fix this:
>
> Search for the problematic add-ons and remove them.
> Visit the Troubleshooting Information page.
> In the Add-ons section, search for one of the following entries:
> Name: Bypass
> ID: {7c3a8b88-4dc9-4487-b7f9-736b5f38b957}
> Name: Bypass XM
> ID: {d61552ef-e2a6-4fb5-bf67-8990f0014957}
> Please make sure the ID matches exactly as there might be other, unrelated add-ons using those or similar names. If none of those IDs are shown in the list, you are not affected. If you find a match, follow these instructions to remove the add-on(s).
>
> https://support.mozilla.org/en-US/kb/disable-or-remove-add-ons
>
> Try refreshing Firefox, which will reset your add-ons and settings.
> You can also head over to download a new copy of Firefox if needed.

So just a heads-up. Since 455,000 Firefox users have downloaded and installed this "Bypass" or "Bypass XM" add-ons (whatever they do) and since our audience might tend to be in the demographic that would go for something like that, I wanted to be sure everyone knows of this. It's not the end of the world, but everyone will be wanting to keep their browsers up to day and these add-ons prevent that, apparently deliberately.

Windows 11 News

Before I start talking about the state of Windows yet again, today, I want to acknowledge that I **am** feeling self conscious about that fact that I've been pounding on Microsoft, to the extent that I have, pretty much since March of this year, when we learned that they hadn't bothered to fix the Exchange Server problems they'd been informed of many months earlier. And that they apparently only finally did so — and even then incorrectly, incompletely and incompetently — when their own customers began coming under attack as a consequence of their protracted negligence.

This podcast has been observing and reporting the facts as they have unfolded, and I don't know what else I can do. I also can't help but have an opinion and editorialize a bit because I've been programming computers since I was 16 years old in High School 40 years ago. So I can state with absolute authority that while these machines are fascinatingly complex, they are not mystical or magical or unknowable. It's called "computer science" not "computer alchemy."

As we will see shortly, Microsoft has begun selling the idea that this is beyond us all. That no one is in control, that no one **can** be in control any longer. That along with them, all we can do is hope for the best. I refuse to be suckered into that mindset, and I assume that this podcast's listeners are with me. Every single one of us is here because we love and are fascinated by computers, and because we want to understand them. They are deterministic and knowable. They serve us, not the other way around. I'm truly worried that Windows appears to have escaped Microsoft's grasp. If that's the case, acknowledging and examining that truth is worth our while.

So... Can we print yet? Is that too much to ask?
On the one hand, no one is going to believe this, or want to believe this. But on the other hand, unbelievable as it is, in a sad way it won't come as a shock...

Microsoft has said that after installing October's Patch Tuesday updates — KB5006670 for Windows 10 and KB5006674 for Windows 11 — customers have been experiencing issues with... wait for it... network printing. Microsoft explained that Windows users attempting to connect to networked shared printers might encounter multiple errors preventing them from printing over the network. For example, after deploying KB5006674 for Windows 11, the errors generated can be:

- 0x000006E4 (RPC_S_CANNOT_SUPPORT)
- 0x0000007C (ERROR_INVALID_LEVEL)
- 0x00000709 (ERROR_INVALID_PRINTER_NAME)

I surveyed the complete list of Windows platforms affected and I didn't see any that were missing... from the most recent all the way back to and including Windows 7 Service Pack 1. So here's an instance where the 15% of users whose Windows 7 have stopped receiving "improvements" from Microsoft can feel glad.

So, yes, for the past two weeks, ever since installing last month's latest "improvements" for Windows, Win10 admins and their users have been reporting widespread network printing problems. Over on BleepingComputer's forum, the postings extend for 14 pages. And, once again, while the posters have told the tales of their frustrating and frustrated attempts to deal with the new printing problems, they once again came to the previous month's conclusion: uninstalling the October cumulative updates resolves the printing problem.

In fact, since then, the issues have grown so severe that Windows admins have reportedly resorted to taking matters into their own hands, replacing Windows DLLs with older versions to re-enable printing. BleepComputer reports that the DLLs admins are replacing to get their enterprise's printing working again are **localspl.dll**, **win32spl.dll**, and **spoolsv.exe**. And while stepping back to earlier DLL's will remove Microsoft's multiple attempted fixes for the various PrintNightmares, since Microsoft continues to be unable to get their network printing to work **with** security, it's either go without security and print, or go without printing.

Also, manually replacing only those three DLL culprits has the benefit of leaving the rest of October's updates in place, which would otherwise all need to be rolled back as a whole.

Next Tuesday, Microsoft will have the opportunity to take another shot at it. It does feel as though they are finally zeroing in on the trouble. But it sure also feels as though their entire user

base has become their field test lab. You'd think that they would have some well-established testing facilities for checking these things before subjecting the rest of the world to their repeated and failed experiments. But, on the other hand, we know that they have lost exactly zero customers as a result of this incompetence. So why would they bother? I know how harsh that sounds, but I don't see another rational explanation. We might not like that explanation, but it is brutally rational.

A new Local Privilege Escalation affecting all versions of Windows

A security researcher by the name of "Abdelhamid Naceri" has disclosed technical details for a Windows privilege elevation vulnerability and a public proof-of-concept (PoC) exploit that gives SYSTEM privileges under certain conditions. Normally, this would be regarded as a hostile act. But in this case it's difficult to blame Naceri for posting what he has.

Unfortunately, this appears to be another instance of Microsoft treating a symptom and not the cause. Back in August, Microsoft released a security update titled: "Windows User Profile Service Elevation of Privilege Vulnerability" and formally tracked as CVE-2021-34484. That bug had been discovered by none other than Abdelhamid Naceri.

Once his discovery had been remediated, Naceri was understandably curious to see how Microsoft had fixed the glitch he had found. But upon examining the updated code, he discovered that the patch didn't fix the actual problem at all and that he was still easily able to bypass it with another slightly altered exploit, which is what he has published on GitHub: https://github.com/klinix5/ProfSvcLPE/raw/main/write-up.docx

The technical details of the vulnerability require some understanding of the Windows API and its internals to be meaningful. But the gist of the issue is, as I said, that we have another instance of someone at Microsoft who's responsible for examining, understanding and fixing these bugs, instead tweaking Windows' code to keep a security researcher's provided Proof-Of-Concept from functioning rather than truly examining, understanding and repairing the underlying problem for which the PoC was simply one possible instance of exploitation.

For what it's worth, here's what Naceri wrote:

> *"Technically, in the previous report CVE-2021-34484 I described a bug where you can abuse the user profile service to create a second junction. But as I see from ZDI advisory and Microsoft patch, the bug was metered as an arbitrary directory deletion bug. Microsoft didn't patch what was provided in the report but the impact of the PoC. Since the PoC I wrote before was horrible, it could only reproduce a directory deletion bug. As from the quick patch analysis, they didn't do any major changes to the code. They only removed the CDirectoryRemover destructor which remove the directory. Unfortunately, this isn't sufficient to fix the bug."*

The revised PoC will cause an elevated command prompt with SYSTEM privileges to be launched while the User Account Control (UAC) prompt is displayed. CERT/CC's vulnerability analyst, Will Dormann, tested the vulnerability and found that while it worked, it was temperamental and did not always create the elevated command prompt. But BleepingComputer tested the vulnerability and it launched an elevated command prompt immediately and successfully.

The good news is that the exploitation of the bug requires a threat actor to have the login credentials for some other account on the system. But there are scenarios where this could allow a user to escape administrative control. And Dormann agreed that it is "Definitely still a problem. And there may be scenarios where it can be abused. But the two-account requirement probably puts it in the boat of not being something that will have widespread use in the wild." And I certainly agree with that. However, Naceri told BleepingComputer that a threat actor only needs another domain account to exploit the vulnerability, so it should still be something to be concerned about. And for what it's worth, Microsoft said they are aware of the issue and are looking into it.

My greater concern is that Windows has always been buggy, but it's also always been workable. But it feels as though Microsoft is starting to treat outside security researchers as an annoyance to be muted as quickly as possible rather than as a source of vital and irreplaceable security information that can serve to make their operating system better.

Ask your AI

In a announcement accompanying an update to their Windows PC Health Check app (which we'll get to in a minute) Microsoft said that as part of their phased rollout of Windows 11 to existing Windows 10 users... listen to this:

"The availability of Windows 11 has been increased and we are leveraging our latest generation machine learning model to offer the upgrade to an expanded set of eligible devices."

And according to previous statements, Microsoft estimates that all eligible Windows 10 devices will be offered the upgrade to the latest version by mid-2022.

Now, **nothing** about that should sound reasonable. In fact, remembering that computing is entirely deterministic, it's difficult to imagine it's even true. So Microsoft... you currently have Windows 10 working on **all** PCs everywhere in the world. It simply runs, anywhere. No problem. And many years ago before Windows 10, although not everyone wanted Win10 and you had to force many people to take it, anyone who had Windows 7 or 8 could immediately upgrade to Windows 10 when it became available and everything worked just fine.

But now, you've so dramatically advanced the state of the art that you no longer understand your own operating system. If we're to believe what you're saying, it's become a mystery to you. It's gotten away from you. And instead of it running better and in more places, you no longer know where, or if, it will run at all. So you've built yourselves a machine learning model; an AI; an oracle to tell **you** where it might be safe to give it a try... based on what? Statistics? Hope? A complex statistical model that you need an AI to peer into? What — has — gone — wrong?

You are [and I quote] *"leveraging [y]our latest generation machine learning model to offer the upgrade to an expanded set of eligible devices."* So, you've somehow taken an operating system which could, for many generations, install and run on any Intel-based hardware. But now it's so advanced that it can no longer run on all Intel-based hardware—only some—but you don't know where, what or which. As I said, nothing about any of this should sound reasonable. Something has gone very wrong in Redmond.

And speaking of the PC Health Check
We've also learned that as of Friday, Microsoft has begun to force-install their PC Health Check application — the one in this week's picture of the week — onto Windows 10 devices using a Windows Update KB5005463.

Windows 10 users quickly noticed this and began complaining to Microsoft. Microsoft said that any users who do not want PC Health Check on their system can simply uninstall it using the Settings app. However those who have done so numerous times have discovered that it will be seen as missing and will be reinstalled during the system's next check for updates. And even more maddeningly, when attempting to uninstall KB5005463, Windows 10 may inform its user that the update is not installed when the user is looking right at it.

There's a registry key named “PreviousUninstall” which is supposed to be set to indicate that a user has manually removed an update so that Windows Update will not see that the update is missing and reinstall it. But for some reason that mechanism isn't working reliably in this case.

And there's no real reason not to want the PC Health Check. It appears to be mostly users who are affronted by yet another indignity foisted upon them by Microsoft. Old timers still haven't given up imagining that they control their own machines. It's becoming clear that the only way for that to be true will be to move to Linux. Linus Torvald's dream of building a truly attractive and personal operating system is being more fully realized every day.

Security News

Stand back for the Adobe Security Patch Tsunami
Out of 92 security vulnerabilities, 66 are rated CRITICAL severity, most allowing for code execution and many for private information disclosure. This month's Patch Tuesday is next Tuesday and apparently Abode felt that this couldn't wait, even though it did draw a great deal of industry attention.

Threatpost wrote “Adobe has dropped a mammoth out-of-band security update this week, addressing 92 vulnerabilities across 14 products.” and ZDNet's headline was “Weeks early: Adobe dumps massive security patch update.”

Okay, so what do we know? As I said, the majority of the bugs, nearly 2/3rds, are rated critical with most of those allow arbitrary code execution. But we also have escalation of privilege, denial-of-service and memory leak information disclosure. And these are spread liberally across 14 different Adobe properties: After Effects, Animate, Audition, Bridge, Character Animator, Illustrator, InDesign, Lightroom Classic, Media Encoder, Photoshop, Prelude, Premiere Pro, Premiere Elements and the XMP Toolkit SDK.

Most of the bugs were reported by just two teams: The TopSec Alpha Team and Trend Micro’s Zero-Day Initiative (ZDI).

Dustin Childs of ZDI told Threatpost that “Of the patches released by Adobe, nine of these came through the ZDI program. Most of these are simple file-parsing bugs, but there are a couple of critical-rated out-of-bounds (OOB) write bugs as well. For these, the vulnerability results from

the lack of proper validation of user-supplied data, which can result in a write past the end of an allocated structure. An attacker can leverage these bugs to execute code in the context of the current process."

What's also somewhat bracing is that Adobe had just, two weeks earlier, released their regularly scheduled monthly Patch Tuesday. And despite the out-of-cycle nature of these 92 new vulnerability patches, none are 0-days having evidence of any active exploitation in the wild.

So, for what it's worth, you know if you use any Abode products. So it might be prudent to check-in with them for any updates.

The VoIP DDoS attacks continue

A UK based VoIP industry group representing the UK telecommunications sector said last week that several of its members active in the Voice-over-IP (VoIP) market had been hit by Distributed Denial of Service (DDoS) attacks over the past month. We have talked about some of these recent attacks previously. In a statement last Tuesday, the Comms Council UK said the DDoS attacks were "part of a coordinated extortion-focused international campaign by professional cyber criminals."

The organization did not share the name of the victims, but VoIP providers like Voipfone, VoIP Unlimited, and VoIP.ms have previously disclosed that they were the subject of DDoS extortion attempts since the end of August. And in addition, Bandwidth.com, an upstream provider for many VoIP companies, said it was also attacked as part of this extortion campaign, which the company said it managed to mitigate at the end of September.

The threat actors first launched DDoS attacks, then sent eMail demanding huge payouts to stop the attacks. They knew that companies such as VoIP providers could not afford to remain offline without incurring huge financial losses and pressure from their customers. And as we covered at the time, attacks against a VoIP's bandwidth and infrastructure, being other than DNS and Web, are significantly more difficult to mitigate.

David Morken, the CEO of Bandwidth.com said earlier last month: "The attackers took advantage of the unique characteristics of real-time communications, as well as the highly interconnected nature of our industry." Cloudflare, which has been helping mitigate these attacks together with other DDoS mitigation providers, has also noted a recent focus on VoIP providers.

But despite the numerous reports and press coverage surrounding this campaign, the attackers have not been discouraged by the media attention. The attacks remain ongoing with Voipfone still dealing with a wave of DDoS attacks that began last Monday, according to the company's server status page.

All the affected companies said the attacks crippled their infrastructure and affected telephony and messaging services for their customers, resulting in prolonged, multi-day outages.

Eli Katz, the Chair of Comms Council UK said the attacks had extensive secondary impacts upon "critical infrastructure organisations including the Police, NHS and other public services." He described the DDoS extortion campaign as "attacks on the foundations of UK infrastructure."

Coordinated DDoS attacks against selected industry sectors have occurred before, and they appear to focus on industries that can't afford to go offline, even for a few minutes. We've talked about the original DDoS attacks against gambling sites which were desperate to remain online during major sporting events. Similar attacks occurred a year ago, in September 2020, when attackers launched a similar campaign against EU-based internet service providers. Other campaigns have targeted entities in the financial sector, such as banks and stock markets. And another recent sector to be targeted with DDoS extortion campaigns has been privacy and security-focused email providers. Victims of these DDoS attacks, which continued throughout last week included Runbox, Posteo, Fastmail, TheXYZ, Guerilla Mail, Mailfence, Kolab Now, and RiseUp.

We know that the Internet now has many massive botnets which are capable of producing astonishing levels of aggregate bandwidth. And the lack of egress filtering allows their outbound packets to carry any IP address they choose. A long time ago this podcast looked at the incredibly elegant simplicity of the autonomous packet routing invention. Unfortunately, for all it's elegant simplicity it was never designed to prevent its own abuse. And so today we have evermore attack targets hiding behind the services of CloudFlare and other attack mitigation services.

Closing The Loop

Dave N / @RandomDaveInFL
Hello Steve!!! Just listened to the latest podcast (I'm a bit late in listening this week) and you probably already know about "Lasguns" in Dune. [I didn't, so thank you, Dave.] But there actually is an answer why they don't use them more often. The interaction with the Lasgun and shield is catastrophic for everyone. They mention this in the book but never really mentioned it in the movie. I've been a fan of Dune for many many years and glad to finally see a movie that does the books some justice. Can't wait till part 2. Only wish it was sooner. Have a great day!!!!!

https://dune.fandom.com/wiki/Lasgun

> *"Lasguns were the preferred weapon for armies. However, when shields were being employed, lasguns were generally not used because contact reaction between a lasgun beam and a shield created a nuclear explosion that often killed everyone within a large radius.*
>
> *Many soldiers and assassins preferred knives and swords in combat, both because they safely penetrated personal shields, and because of newfound appreciation for the art of swordsmanship."*

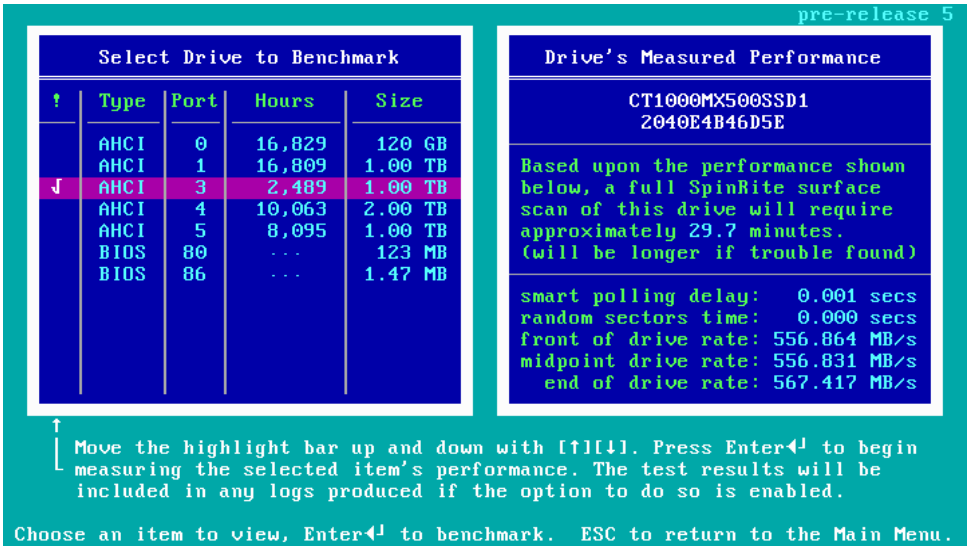
Peter G. Chase / @PchaseG
I'll be reading a newsletter and get the overlay asking me if I want to sign up for their newsletter... which is how I got to that page in the first place, because I'm signed up.

jdainsworth / @jdainsworth
At Microsoft PM is usually Project Manager or Program Manager, and S is either senior or Security. SPM is most likely: "Security Program Manager."

SpinRite

Sunday afternoon at 4:00 PM I posted SpinRite v6.1's 5th pre-release. The gang who were waiting for it wasted no time copying the 55K DOS executable to their bootable thumb drives and taking it out for a spin. They found the almost entirely re-written benchmarking system with the new feature I added: Once SpinRite has benchmarked a drive, it estimates and displays the total time that will be required for SpinRite to perform a standard Level 2 analytical scan of that drive's entire storage space. The best news is how fast and practical SpinRite v6.1 is turning out to be. SpinRite v6.1 will be able to perform a scan of...

A 1TB Crucial SATA SSD in 29.7 minutes.
One terabyte in half an hour! (See screenshot below.) As I've mentioned previously, I was originally hoping for half a terabyte per hour from the new SpinRite. This is four times faster than that. It's a new land speed record for SpinRite!
A 256 GB SATA SSD in 7.3 minutes.
A Kingston 129 GB SATA SSD in 3.7 minutes.
A 6TB spinning Seagate SATA drive in 9.27 hours.
An old (spinning) 40 GB Maxtor IDE (parallel cable) in 31.1 minutes.
An old (spinning) 160 GB Seagate IDE (parallel cable) in 48.1 minutes.
A 64 GB SanDisk USB-attached thumb drive in 59.3 minutes.
A 1 TB USB-attached solid state drive in 14.46 hours.



We won't have SpinRite's new super-speed performance for USB until v7.1, since I'm going to need to write USB drivers from scratch as I just have for IDE, ATA and AHCI controllers. And since the next step will be to move SpinRite away from DOS to a new 32-bit OS so that it will be able to boot on UEFI-only systems, it was much more efficient to wait to add USB support until we're there to minimize rewriting. But even 14.46 hours for a 1TB drive is significantly more practical than SpinRite's previous performance.

Aside from providing lots of fun benchmarks, although the 5th pre-release resolved a bunch of previous edge cases, it also quickly revealed a few issues with specific hardware where I'll be spending some more time. But, overall, things are looking very good!

“Trojan Source”

A pair of researchers in the UK have been exceedingly clever. They have figured out an entirely practical and terrifyingly effective way of hiding malicious source code in plain sight. Or if not really in plain sight then invisibly right in front of everyone else's eyes without being seen.

Their 15-page research paper Abstract reads:

> *We present a new type of attack in which source code is maliciously encoded so that it appears different to a compiler than to the human eye. This attack exploits subtleties in text-encoding standards such as Unicode to produce source code whose tokens are logically encoded in a different order from the one in which they are displayed, leading to vulnerabilities that cannot be perceived directly by human code reviewers. ‘Trojan Source’ attacks, as we call them, pose an immediate threat both to first-party software and of supply-chain compromise across the industry. We present working examples of Trojan-Source attacks in C, C++, C#, JavaScript, Java, Rust, Go, and Python. We propose definitive compiler-level defenses, and describe other mitigating controls that can be deployed in editors, repositories, and build pipelines while compilers are upgraded to block this attack.*

So what, exactly, have these clever guys come up with? They introduce their ideas quite well, so I'm going to share their introduction:

What if it were possible to trick compilers into emitting binaries that did not match the logic visible in source code? We demonstrate that this is not only possible for a broad class of modern compilers, but easily exploitable.

We show that subtleties of modern expressive text encodings, such as Unicode, can be used to craft source code that **appears** visually different to developers than to compilers. The difference can be exploited to invisibly alter the logic in an application and introduce targeted vulnerabilities.

The belief that trustworthy compilers emit binaries that correctly implement the algorithms defined in source code is a foundational assumption of software. It is well-known that malicious compilers can produce binaries containing vulnerabilities; as a result, there has been significant effort devoted to verifying compilers and mitigating their exploitable side-effects. However, to our knowledge, producing vulnerable binaries via unmodified compilers by manipulating the encoding of otherwise non-malicious source code has not so far been explored.

Consider a supply-chain attacker who seeks to inject vulnerabilities into software upstream of the ultimate targets, as happened in the recent Solar Winds incident. Two methods an adversary may use to accomplish such a goal are: Suborning an insider to commit vulnerable code into software systems, and contributing subtle vulnerabilities into open-source projects. In order to prevent or mitigate such attacks, it is essential for developers to perform at least one code or security review of every submitted contribution. However, this critical control may be bypassed if the vulnerabilities do not **appear** in the source code displayed to the reviewer, but are hidden in the encoding layer underneath.

Such an attack is quite feasible, as we will hereafter demonstrate. In this paper, we make the following contributions.

- We define a novel class of vulnerabilities, which we call Trojan-Source attacks, and which use maliciously encoded but semantically permissible source code modifications to introduce invisible software vulnerabilities.

- We provide working examples of Trojan-Source vulnerabilities in C, C++, C#, JavaScript, Java, Rust, Go, and Python.

- We describe effective defenses that must be employed by compilers, as well as other defenses that can be used in editors, repositories, and build pipelines.

- We document the coordinated disclosure process we used to disclose this vulnerability across the industry.

- We raise a new question about what it means for a compiler to be trustworthy.

We're familiar with how the expansive UNICODE character set can be, and has been, used to create so-called "homograph attacks" using lookalike or closely alike characters in a domain name. A user might click on a link, then look in their browser's URL field to verify where they are before proceeding to interact with the website. But in a homograph attack, the domain they are actually on looks like "PayPal" but isn't. The way they state this more formally is:

Digital text is stored as an encoded sequence of numerical values, or code points, that correspond with visual glyphs according to the relevant specification. While single-script specifications such as ASCII were historically prevalent, modern text encodings have standardized around Unicode.

At the time of writing, Unicode defines 143,859 characters across 154 different scripts in addition to various non-script character sets (such as emojis) plus a plethora of control characters. While its specification provides a mapping from numerical code points to characters, the binary representation of those code points is determined by which of various encodings is used, with one of the most common being UTF-8.

Text rendering is performed by interpreting encoded bytes as numerical code points according to the chosen encoding, then looking up the characters in the relevant specification, then resolving all control characters, and finally displaying the glyphs provided for each character in the chosen font.

Not all languages are read from left to right. And Unicode's textual representation was designed to support all languages. This means that there must be some means for controlling the reading direction — the visual glyph positioning — through Unicode.

In order to support left-to-right and right-to-left ordered languages, Unicode defines a set of nine control characters such as "RLE" which stands for right-to-left embedding, RLO for "right-to-left override". And since these are locally modal, it's necessary to have some way of undoing them. So there's a "PDF" character which stands for "Pop Directional Formatting" —

literally like popping a stack. And the definition states that it "Terminate nearest LRE, RLE, LRO, or RLO." So that means that Unicode rendering is an interpreter which is maintaining state.

In explaining their attack methodology, they write:

Internationalized text encodings require support for both left-to-right languages such as English and Russian, and right-to-left languages such as Hebrew and Arabic. When mixing scripts with different display orders, there must be a deterministic way to resolve conflicting directionality. For Unicode, this is implemented in the Bidirectional, or Bidi, Algorithm.

In some scenarios, the default ordering set by the Bidi Algorithm may not be sufficient; for these cases, override control characters are provided. Bidi overrides are invisible characters that enable switching the display ordering of groups of characters.

The table provides a list of Bidi override characters relevant to this attack. Of note are LRI and RLI, which format subsequent text as left-to-right and right-to-left respectively, and are both closed by PDI. [Since our listeners cannot see the table, LRI stands for left-to-right isolate, RLI stands for right-to-left isolate, and PDI stands for pop directional isolate. We'll get to "isolates" in a minute.]

Bidi overrides enable even single-script characters to be displayed in an order different from their logical encoding. This fact has previously been exploited to disguise the file extensions of malware disseminated by email and to craft adversarial examples for NLP machine-learning pipelines.

As an example, consider the following Unicode character sequence: RLI a b c PDI which will be displayed as: c b a

All Unicode Bidi overrides are restricted to affecting a single paragraph, as a newline character will explicitly close any unbalanced overrides, namely overrides that lack a corresponding closing character.

And to the ability to reorder individual character sequences, there's one additional complexity which adds to the ability to use tricky Unicode encodings, and that's those isolates: In the Bidi specification, isolates are groups of characters that are treated as a single entity; that is, the entire isolate will be moved as a single block when the display order is overridden. And, moreover, isolates can be nested.

So, {RLI} **{LRI} abc {PDI} {LRI} def {PDI}** {PDI} will display as: defabc. Meaning that the two linear runs of characters "abc" and "def" have been taken as a group and exchanged.

So by this point everyone gets it, right? They write:

Embedding multiple layers of LRI and RLI within each other enables the near-arbitrary reordering of strings. This gives an adversary fine-grained control, so they can manipulate the display order of text into an anagram of its logically-encoded order.

Like most non-text rendering systems, compilers and interpreters do not typically process

formatting control characters, including Bidi overrides, prior to parsing source code. This can be used to engineer a targeted gap between the visually-rendered source code as seen by a human eye, and the raw bytes of the encoded source code as evaluated by a compiler.

We can exploit this gap to create adversarially-encoded text that is understood differently by human reviewers and by compilers.

And they did it. They pulled it off. They successfully implemented invisible malicious changes to the source code of C, C++, C#, JavaScript, Java, Rust, Go, and Python. In their paper they take each language at a time and show how these techniques can be used in the real world.

So we have a new and entirely viable and attractive means of attacking the source code of pretty much everything. This should worry us. So I went looking for and found their discussion of who they told about this before they told all of the bad guys throughout the world. They wrote:

We contacted nineteen independent companies and organizations in a coordinated disclosure effort to build defenses for affected compilers, interpreters, code editors, and code repository front-ends. We set a 99-day embargoed disclosure period during which disclosure recipients could implement defenses before we published our attacks. We met a variety of responses ranging from patching commitments and bug bounties to quick dismissal and references to legal policies.

We selected an initial set of disclosure recipients by identifying the maintainers of products that our experiments indicated were affected by the Trojan Source vulnerability pattern. We also included companies that, to our knowledge, maintained their own internal compilers and build tools. The initial disclosures were sent on July 25, 2021.

Several of the initial recipients asked us to include additional organizations in the disclosure process, and we did so. We also sent additional disclosures throughout the embargo window for affected products that we discovered during the disclosure process.

Of the nineteen software suppliers with whom we engaged, seven used an outsourced platform for receiving vulnerability disclosures, six had dedicated web portals for vulnerability disclosures, four accepted disclosures via PGP-encrypted email, and two accepted disclosures only via non-PGP email. They all confirmed receipt of our disclosure, and ultimately nine of them committed to releasing a patch.

Eleven of the recipients had bug bounty programs offering payment for vulnerability disclosures. Of these, five paid bounties, with an average payment of $2,246.40 and a range of $4,475.

On September 9, 2021, we sent a vulnerability report to CERT/CC, the CERT Coordination Center sponsored by CISA. Our report was accepted the same day for coordinated disclosure assistance. This gave all affected vendors access to VINCE, a tool providing a shared communication platform across vendors implementing defenses. Thirteen of our recipients, inclusive of CERT/CC, opted in to the VINCE tool for these shared communications. CERT/CC also added three additional vendors to the disclosure beyond the nineteen we had already contacted.

On October 18, 2021, Trojan Source attacks were issued two CVEs: CVE-2021-42574 for

tracking the Bidi attack, and CVE-2021-42694 for tracking the homoglyph attack. These CVEs were issued by MITRE against the Unicode specification.

On the same day, we sent a PGP-encrypted disclosure to the distros mailing list [44], which contains representatives of the security teams of 21 operating systems as of the time of writing. This list coordinates the application of patches across OS maintainers, but allows a maximum embargo period of 14 days.

And, finally, get this...

We were curious if we could find any examples of Trojan Source attacks in the wild prior to public disclosure of the attack vector, and therefore tried to scan as much of the open source ecosystem as we could for signs of attack.

We assembled a regex that identified unterminated Bidi override sequences in comments and strings, and GitHub provided us with the results of this pattern run against all public commits containing non-markup language source code ingested into GitHub from January through mid October 2021. This yielded 7,444 commits, which resolved to 2,096 unique files still present in public repositories as of October 2021.

The majority of the results were false positives. Examples of clearly non-malicious encodings included LRE characters placed at the start of file paths, malformed strings in genuinely right-to-left languages, and Bidi characters placed into localized format string patterns.

However, we did find some evidence of techniques similar to Trojan Source attacks being exploited. In one instance, a static code analysis tool for smart contracts, Slither, contained scanning for right-to-left override characters. The tool provides an example of why this scan is necessary: it uses an RLO character to swap the display order of two single-character variables passed as arguments. In another instance, we discovered the use of RLI and LRI characters used to conceal an invocation of **system("cat /etc/passwd");** within a Ruby script.

In other words, these techniques **had** already occurred to someone and **had** already been under exploitation in the field. They also have some very good thoughts about the implementation of defenses for this class of attacks:

The simplest defense is to ban the use of text directionality control characters both in language specifications and in compilers implementing these languages.

In most settings, this simple solution may well be sufficient. If an application wishes to print text that requires Bidi overrides, developers can generate those characters using escape sequences rather than embedding potentially dangerous characters into source code.

This simple defense can be improved by adding a small amount of nuance. By banning all directionality-control characters, users with legitimate Bidi-override use cases in comments are penalized. Therefore, a better defense might be to ban the use of unterminated Bidi override characters within string literals and comments. By ensuring that each override is terminated – that is, for example, that every LRI has a matching PDI – it becomes impossible to distort legitimate source code outside of string literals and comments.

Trojan-Source defenses must be enabled by default on all compilers that support Unicode input, and turning off the defenses should only be permitted when a dedicated suppression flag is passed.

While changes to language specifications and compilers are ideal solutions, there is an immediate need for existing code bases to be protected against this family of attacks. Moreover, some languages or compilers may choose not to implement appropriate defenses. To protect organizations that rely on them, defenses can be employed in build pipelines, code repositories, and text editors.

Build pipelines, such as those used by software producers to build and sign production code, can scan for the presence of Bidi overrides before initiating each build and break the build if such a character is found in source code. Alternatively, build pipelines can scan for the more nuanced set of unterminated Bidi overrides. Such tactics provide an immediate and robust defense for existing software maintainers.

Code repository systems and text editors can also help prevent Trojan-Source attacks by making them visible to human reviewers. For example, code repository front-ends, such as web UIs for viewing committed code, can choose to represent Bidi overrides as visible tokens, thus making attacks visible, and by adding a visual warning to the affected lines of code.

Code editors can employ similar tactics. In fact, some already do; vim, for example, defaults to showing Bidi overrides as numerical code points rather than applying the Bidi algorithm. However, many common code editors do not adopt this behavior, including most GUI editors such as, at the time of writing, Microsoft's VS Code and Apple's Xcode.

https://trojansource.codes/trojan-source.pdf