Trojan Source

Description: This week we keep counting them Chrome zero-days. We look at a pair of badly misbehaving Firefox add-ons with Mozilla's moves to deal with them 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 mother lode of critical updates from Adobe and at the still-ongoing DDoS attacks against VoIP providers and their providers. We 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. We finish by looking at a frighteningly clever and haunting new attack against source code known as "Trojan Source."

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-843.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-843-lq.mp3

SHOW TEASE: It's time for Security Now!. I'm Jason Howell filling in for Leo Laporte. But of course the man, Steve Gibson, is here, and he's got a lot to talk about. He's going to dive into how Windows 11 keeps disappointing him. Number of facets there that he dives into. Also a large number of Chrome zero-days and what's behind that. A closer look at a new attack that's targeting source code. That's something called "Trojan Source." And a whole lot more coming up next on Security Now!.

JASON HOWELL: This is Security Now! with Steve Gibson, Episode 843, recorded on Tuesday, November 2nd, 2021: Trojan Source.

It's time for Security Now!. I'm Jason Howell filling in for Leo Laporte, who is, well, he's in Mexico, enjoying the last, unfortunately the last couple of days of his trip to Mexico with Lisa. This is the show where we don't talk about taking trips and vacations around the world. We talk about security, the latest security news, with none other than the man you're always arriving for, Steve Gibson. How's it going, Steve?

Steve Gibson: Hey, everybody. And Jason, great to be with you today. Has anyone heard anything from Leo? Have there been texts and pictures and postings and so forth? I haven't been following.

JASON: I mean, little bleeps and bloops, yeah, here and there on Twitter. But you know, primarily any of the feedback that I'm finding about their trip isn't coming from Leo or Lisa, it's coming from Mike Elgan, who is kind of leading the experience that they're there for.

Steve: Ah, the gastronomic nomad.

JASON: Yes. That's the Gastronomad, exactly. And so he's been, Mike's been sharing photos of the Day of the Dead, which they just experienced. Or am I thinking of that horror movie?

Steve: And did he take your camera? Did he take your phone with him?

JASON: Yeah. Yeah, yeah. He took the Pixel 6 Pro. So I'm actually kind of surprised that I haven't seen a bunch of photos. But maybe he's so engulfed in his vacation that connecting to the Internet and making time to share things is off his radar, which let's be honest, when you take a vacation...

Steve: Yes, he's trying to get away from it all. Get away from it all.

JASON: Yeah, exactly. It's a good thing. It's a good thing.

Steve: Okay. So we've got Episode 843 for November 2nd, the first episode of November. I had already selected this as the topic for today when my Twitter feed blew up with people saying, "Oh, my god, what?" Anyway, it's called Trojan Source, which is a frighteningly clever and appropriate for Halloween haunting new attack against source code, which we're going to have fun talking about because it's a couple of researchers in the U.K. who sort of thought outside the box and presented something which is really, everyone's going to understand, which is the fun part about it, but also surprisingly powerful. But of course we've got a lot of other things to talk about.

We're going to keep counting them Chrome zero-days. We've got a couple more, and we're still breaking records. We're going to look at a pair of badly misbehaving Firefox add-ons, and Mozilla's moves to deal with not only them, but also future abuse of the thing that they were doing. We check in with Windows. Yes, once again, news from Redmond. And I'm going to be unable to resist commenting on some of the new nonsense that's coming from there. I just, well, I'll save that.

Then we're going to look at a surprise mother lode of critical updates from Adobe, and at the still ongoing DDoS attacks against VoIP providers and their providers. We look at some fun and interesting closing-the-loop feedback from our listeners. And I'm able to share what even surprised me, some early benchmarks from SpinRite, which I did drop that fifth prerelease that I've been talking about for a couple of weeks. And it's going to be really fast.

Anyway, then, as I said, we'll wrap up by taking a look at something that everybody's going to be able to understand. And it's just spooky. And of course we have a great Picture of the Week, which is apropos of some of the things we'll be talking about regarding Microsoft. So I think another great podcast for our listeners.

JASON: Always a great podcast for our listeners. And you mentioned spooky. That's apropos because, well, was it yesterday? No, it was the day before yesterday was Halloween. So you're a few days off from like the spooky, the spookiest of days. But that's okay. We can round things out with that. And now we're going to get into your Picture of the Week. Also has to do with Windows 11. You've got a lot to say about Windows 11 today.

Steve: I have too much to say about Windows 11.

JASON: Always.

Steve: But can't help myself. So this was a screenshot of a desktop. One of our listeners, he tweeted me, his handle is StarKiss, @StarKissedOne, O-N-E. And his tweet said:

"Apparently my Dell Core-i7 is both compatible and incompatible with Windows 11 at the same time." Then he said: "Best quote from last show: 'Win11 is for people who like pain.'"

So anyway, the desktop that he set up and took a picture of, on the left-hand side of the screen we see the PC Health Check app, which we will also be talking about today. And it says, it's all got green checkmarks: "This PC meets Windows 11 requirements." And specifically the third one down says, and you'll see why I'm pointing this out in a second: "The processor is supported for Windows 11. Processor: Intel Core i7-7820HQ CPU @ 2.90GHz."

And then over on the right of the same screen, we've got Windows 11 Setup, which was trying to run. And it reports quite loudly: "This PC doesn't currently meet Windows 11 system requirements. Here's why." And then we've got a black X, rather than the happy green checkmark we had before, saying: "The processor isn't supported for this version of Windows." So, you know...

JASON: Clear as mud.

Steve: They can't make up their mind because the whole thing is BS. But we'll get to that in a minute. First, let's talk about what's going on with Chrome. More zero-days. Two newly discovered critical, true, in-the-wild zero-days, and I highlight that because everyone's now using zero-day for everything, even when they're not. So these are real zero-days, discovered being used in the wild. They were just patched in Chrome's most recent urgent release last Thursday when Google pushed out its emergency update, bringing Chrome to where everybody should now be, 95.0.4638.69.

These two are tracked as CVE-2021-38000 and 38003. They were an - and I love how obscure this is. You know, Google doesn't want to tell us what happened. So we get an insufficient validation of untrusted input in a feature called "Intents," as well as a case of - and this is really good - inappropriate implementation, which pretty much covers everything, in V8, which of course is Chromium's JavaScript and WebAssembly engine. Both discoveries are credited to TAG, that's Google's Threat Analysis Group, which reported their findings to the Chromium team for the first case on September 15th, and then the second case was October 26th.

Now, okay, you know, I'm wondering about those dates. I'll get to that in a second. Anyway, their sparse advisory regarding these simply stated that: "Google is aware that exploits for" - and then these two CVEs - "exist in the wild." But the fact that an actively exploited, in-the-wild zero-day was reported to the Chromium team on September 15th and only patched last Thursday does seem a bit slow for the way they've been operating. They've generally been, you know, I've been giving Google lots of props for how quickly they respond, now typically within a few days. Presumably in this instance they had their reason.

So what we know of that first one, that it was a "insufficient validation of untrusted input." So I would tend to say, so maybe it wasn't a big deal. But if it was being abused in the wild, then it was worth somebody abusing it. So I don't know why they didn't fix it sooner. I mean, even since then there have been other updates to Chrome which didn't get this fix in there. So, you know, obviously they had a reason. We don't know what it is. They're not telling.

But in addition to these two biggies, the other vulnerability of note was they repaired a use-after-free vulnerability in Chrome's web transport component. And that's the bug which was demonstrated and leveraged for the first time at the Tianfu Cup contest held in the middle of last month in China, which we talked about. So as happens with these things, the Pwn2Own or the Tianfu Cup, is after the exploit is used and wins its exploiter

some big bucks, hopefully, then it's disclosed to the affected party, and they patch it, as they did, you know, quickly. Again, much more quickly, as I said, than that 38000 CVE. But who knows?

And since we've been counting, these two latest emergency patches bring Chrome's resolved total while being actively in use zero-day updates to a record 16 zero-days since the start of the year. And it's on the screen right now. I've got it in the show notes. Look at September. Ouch. September was a rough month for Chrome. One, two, three, four, five, six problems in September alone. One in July, two in June, three in April, two in March, one in February, and one last month. So anyway, as I've said, this is the largest number of zero-days ever patched in Chrome during any calendar year since Chrome's debut, which was 13 years ago, in 2008.

And in this case, because they're on top of it, and they fix these things, and they stay fixed, and they fix them right, I don't think this represents a problem with Chrome, per se. I think it represents the dramatic increase in focus that Chrome is receiving now that it is number one. Thirteen years ago, you know, everybody was attacking IE with glee. Today it's Chrome because they're the big guys on the block. And if you want to get the most people, you attack the browser that the most people are using.

I mentioned some problems with Firefox. Two naughty Firefox add-ons have been caught abusing an API available to those extensions. Mozilla's Security Blog titled their posting "Securing the proxy API for Firefox add-ons." And they set things up by, in a little bit of a marketing style, explaining what we all know: "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." Okay, right.

Okay. So then they explain what happened: "In early June," they say, "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 block lists, and updating remotely configured content. In total" - and this is why this is significant - "these add-ons were installed by 455,000 users." So close to half a million users. They said: "This post outlines the steps we've taken to mitigate this issue, as well as provide details of what users should do to check whether they're affected. Developers of add-ons that use the proxy API will find some specific instructions below that are required for future submissions."

And in fact I didn't include the developer side of this. I'll just note that what the instructions were was to explicitly, to developers, to explicitly declare your use of the proxy API to improve Mozilla's ability to approve your add-on for submission and download by Firefox users. They want to know you're saying yes, we're using the proxy API. And here's why. So anyway, the two malicious add-ons Mozilla found were blocked to prevent their future installation by other users. They're published under the names "Bypass" and "Bypass XM." And Mozilla said: "To prevent additional users from being impacted by new add-on submissions misusing the proxy API, we paused on approvals for add-ons that use the proxy API until fixes were available for all users."

So they said: "Starting with Firefox 91.1" - and everybody should now be at 93, so this is a couple ago - "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 subsequently 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' with additional mitigations that's been shipped to both current and older Firefox versions." So essentially, they realized they were sort of in a Catch-22. If you had installed these add-ons which were blocking updates because the

add-on was proxying your connection through Firefox and maliciously preventing you from getting the update to Firefox which would then be aware of these malicious add-ons, you'd never be able to get rid of them. Firefox wouldn't be updated any longer. Users would be insecure. And it was a mess.

So with 91.1, what they did was they said, okay, we're going to give ourselves permission to bypass the proxy API, which they had not been able to bypass until then, so that if they're unable to update Firefox through the proxy, then they're going to say, okay, well, sorry, we're going to bypass the proxy because updating Firefox is more important than doing what some extension add-on has done, if it prevents us from updating. So that's in place now.

Okay. So they then said: "As a Firefox user, what should you do next?" Hopefully you will know if you're using something called Bypass or Bypass XM. And if so, you want to get rid of it. It's malicious. They've blocked it. They're removing it where they can. But there may be situations where your updates to Firefox are being blocked by this add-on. And since it sounds like maybe - I didn't even get into - I looked around for what it was, but it's gone now. So I couldn't even figure out what benefit it was offering. Sounds kind of like something our listeners might be interested in.

So that's why I wanted to bring it to everyone's attention. If you're using Bypass or Bypass XM, then you want it gone. I do have details in the show notes I won't drag everybody else through. But the way of searching for it by ID or by name and where to go to remove it, a link to do so. So I imagine our listeners will know if they're doing any of that and can then pursue it on their own. 455,000 users of Firefox, wouldn't be surprised if that is a few of our listeners, at least.

Okay. Now, take a deep breath, Steve. Before I start talking about the state of Windows yet again today, I just want to take a moment to acknowledge that I am feeling somewhat self-conscious about the fact that I've been pounding on Microsoft to the extent that I have pretty much since March of this year, when we all learned that they hadn't bothered to fix the Exchange Server problems they'd been informed many months earlier. And 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. Okay, I refuse to be suckered into that mindset, and I assume that this podcast's listeners are with me on this. 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. And 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 the first question: Can we print yet? Is printing too much to ask? On the one hand, no one is going to believe this, or want to believe it. 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, you know, last month's just by two days, October's Patch Tuesday updates KB5006670

for Windows 10 and KB5006674 for Windows 11, for which we held out high hopes, customers have been experiencing issues with network printing. Unbelievable.

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, and we've got three of them, hex 6E4, which is RPC_S_CANNOT_SUPPORT; or you might get the 7c error, which is the ERROR_INVALID_LEVEL of something; or 709, ERROR_INVALID_PRINTER_NAME. Even though the printer name is apparently valid, or it was last week, not anymore.

I surveyed the complete list of Windows platforms affected, which were listed along with this news from Microsoft, 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, Windows 2008 SP2, or R2. So here's an instance where the 15% of users who are still using Windows 7 and have stopped receiving those "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 active forum, the postings about this issue 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. Of course, they had also uninstalled the September cumulative updates to again resolve the printing problem, and the August cumulative updates to resolve the printing problem. Otherwise you can't print in these instances.

In fact, since then, the issues have grown so severe that Windows admins have reportedly resorted to taking matters into their own hands because they don't want to be 90 days behind on all of these critical zero-day and other windows problems that August and September and October fixed while seemingly being unable to fix Windows printing. So they are, I'm not kidding, replacing Windows DLLs by hand with older versions of the DLLs to reenable printing. BleepingComputer reports that the DLLs admins are replacing to get their enterprises printing, the networked printing working again, are localspl.dll, win32spl.dll, and spoolsv.exe. They all sound like spooling things.

And while stepping back to earlier DLLs 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, and for that matter September's and August's, updates in place, which would otherwise all need to be rolled back as a whole. Which, you know, people really don't want to do.

Next Tuesday - the 9th, right? - Microsoft will have the opportunity to take another crack at it. It does feel as though they're maybe finally zeroing in on the trouble. Maybe they could just put those older - oh, no, no, they can't because those aren't secure. Okay. Anyway, we'll wait to see what happens. It sure 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 out before subjecting the rest of the world to their repeated and failed experiments, which is what's been happening, now for 90 days.

But on the other hand, we know that they have lost exactly zero customers as a result of this repeated demonstration of incompetence. So why would they bother? I know how harsh that sounds, but I don't see any other rational explanation. We might not like it, but it is at least brutally rational.

JASON: And they're not alone in that practice. I mean, Google comes to mind. Google does this all the time with their products. And many Google users feel the same way; right? Like am I, as a Google fan and Google user, am I just constantly beta testing your products for you?

Steve: Yeah.

JASON: But yet we come back, and we come back; you know?

Steve: Yeah. So one thing Microsoft seems to be doing that I think Google isn't is fixing the symptom rather than the cause. We have a new local privilege escalation which affects all versions of Windows. I mean, like right now. A security researcher by the name of Abdelhamid Naceri has disclosed technical details for a Windows privilege elevation vulnerability and posted a public proof-of-concept exploit that gives system privileges under certain conditions. Now, normally this would be regarded as a hostile act, right, to just like post the news of this. But in this case it's difficult to blame Naceri for posting what he has.

Unfortunately, as I said, this appears to be another instance of Microsoft treating a symptom and not the cause. And we've hit on a number of these because this is another pattern which has unfortunately established itself this year. Back in August, Microsoft released a security update titled "Windows User Profile Service Elevation of Privilege Vulnerability." So an elevation of privilege vulnerability in Windows Profile Service. And it was formally tracked as CVE-2021-34484. That bug had been discovered by none other than this guy, Abdelhamid Naceri.

Once his discovery had been remediated by, you know, being patched, 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. I've got a link to the doc which he published.

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, 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 proof of concept was simply one possible instance of exploitation.

Okay. So for what it's worth, what Naceri wrote, he said: "Technically, in the previous report" - then he cites 2021-34484, the original CVE, he said: "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 he phrased it, "as an arbitrary directory deletion bug. Microsoft didn't patch what was provided in the report, but the impact of the proof of concept. Since the proof of concept I wrote before was horrible," he's saying of his own proof of concept, "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 removed the directory. Unfortunately, this isn't sufficient to fix the bug."

Okay. So the revised proof of concept which he posted will cause an elevated command prompt with system privileges to be launched while the User Account Control prompt is displayed. Okay. This is, again, just the proof of concept. CERT/CC's vulnerability analyst whom we often quote, Will Dormann, tested the vulnerability, that is, this proof of concept, and found that while it worked, it was a little temperamental and didn't always create the elevated command prompt. But BleepingComputer tested the vulnerability as

provided, and it launched an elevated command prompt immediately and successfully. And again, this was just to demonstrate the bug hadn't been fixed.

Okay. Now, the good news is that the exploitation of the bug requires a threat actor to have the login credentials, not only for the account they're currently logged into but some other account on the system. While there are scenarios where this could allow a user to escape administrative control, Dormann agreed that it is "definitely still a problem, and there may be scenarios where it can be abused." He said: "But the two-account requirement probably puts it in the boat," as he phrased it, "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.

Again, my greater concern is that Windows, while it's always been buggy, it's 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. You know, that's what we all want. Microsoft says they want it. Users certainly do. Security researchers certainly do. But it's feeling like researchers are now on the outside just annoying Microsoft.

Okay. In an announcement accompanying an update to their Windows PC Health Check app, and as we saw from the Picture of the Week, looks like it did need to be updated since it was disagreeing with Windows 11 setup - and we'll get to that, there's another issue about that in a minute - Microsoft said that as part of their phased rollout of Windows 11 to existing Windows 10 users, okay, 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 I'll just note that according to previous statements, Microsoft estimates that all eligible Windows 10 devices will be offered the upgrade to the latest version by mid-next year, 2022. Okay, now, nothing about any of that should sound reasonable. In fact, remembering that computing is entirely deterministic, it's difficult to imagine that that'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 apparently 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, Microsoft. 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 your 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 that anyone had. 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 wrong in Redmond.

JASON: Something's rotten in Redmond.

Steve: And speaking of that - just it's nuts, Jason. Really. I mean, it's like we're all...

JASON: That is. I love how you put it.

Steve: We're all buying this bullshit. I mean, it is - it's nuts. It's like, oh, well, maybe I'll get lucky, and Windows 11 will run. You know, I didn't want 10. They forced me to have it, and it ran just fine. Now I can't have it. So what, am I supposed to want it, you know, to get the rounded corners so I don't, you know, cut myself on a sharp edge? Oh, it's just looney tunes. You know? And everyone's like, oh, wow, they're using advanced learning model AI to figure out if I can have it or not. What?

JASON: Don't worry. The next version will be better. The next version will win you over again. It's the on-again, off-again TikTok nature of these releases here. All you can do is wait for the next release.

Steve: You know. And let's not forget, let's not forget they couldn't name it Windows 9 because some things might have been confused, I'm not kidding, and thought this was 95 or 98. They were worried about the digit "9" in the version. And it's like, oh, things will break if we call it Windows 9. I mean, that should have been our first clue that there was something wrong.

Wow. Okay. Now, on top of all this, we've also just learned as of last Friday that Microsoft has begun to force install their PC Health Check app, the one in this week's Picture of the Week, which disagrees with Windows 11 setup, onto Windows 10 devices using a Windows update. It's 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, which was snuck in under the guise of Windows Update, 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 by Windows after they uninstall it, and it'll 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. You know, like I said, it's gotten out of control. It's like the inmates are running the asylum.

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. Gee. Imagine that. And there's no reason not to want the PC Health Check; right? I mean, it doesn't run automatically. They're giving it to you so that you can check in, I guess, with the latest machine learning AI to see if it's decided that you can, oh look, now your machine works with it. Apparently it didn't before. We're not sure, but the AI, the oracle that we're consulting seems to think now you can give it a try. If when you actually try to run Windows 11 setup, it lets you. Who knows?

On the other hand, it appears to be mostly users who are affronted by yet another indignity foisted upon them by Microsoft. You know, we old-timers still haven't given up imagining that we control our own machines. It's becoming clear that the only way for that to be true will be to move to Linux. Linus Torvalds' dream of building a truly attractive and personal operating system is being more fully realized every day.

And with that, Jason, I'm going to take a sip of water, and you're going to give us a nice pause.

JASON: If you could just work on, during this break, formulating your ideas around how you really feel, that would be great. We want to know how you really feel, Steve.

Steve: You don't want me to hold back, is that - don't hold back?

JASON: Don't hold back.

Steve: Okay. Okay.

JASON: There's other news to report.

Steve: I know that our listeners are glad for that. But, boy. I just - we need a reality check every so often. This is just, you know...

JASON: For sure.

Steve: Microsoft has deployed a machine learning model, their latest one, you know, we don't want the older one apparently, the latest one to let us know or to let them know what it runs on. It's like, what? It's just nuts. Okay. Anyway, you did try to break me from this.

Let's talk about Adobe. Out of 92 security vulnerabilities, 66 are rated critical in severity, most allowing for code execution, and many for private information disclosure. This month's Patch Tuesday is next Tuesday. But apparently Adobe felt that this couldn't wait, even though it did draw a great deal of industry attention by doing this out of cycle. 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 two thirds of them, are rated critical, with most of those allowing for arbitrary code execution. But we also have escalation of privilege, denial of service, and memory leak information disclosures. 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. Interesting, I didn't see Reader in there. That's kind of amazing. Maybe they fixed that the previous week on October's Patch Tuesday because they did another patch dump then.

Anyway, most of the bugs were reported by just two teams, the TopSec Alpha Team and Trend Micro's Zero-Day Initiative that we referred to a second ago, 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 were simple file-parsing bugs, but there were a couple critical-rated out-of-bounds 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." We know that's not good.

What's also somewhat bracing is that Adobe had just, as I said, two weeks earlier released their regularly scheduled monthly Patch Tuesday. And despite the out-of-cycle nature of these 92 new additional vulnerability patches, none - and this is the good news - none are zero-days having any evidence of active exploitation in the wild. So maybe they just figured they'd get ahead of the curve. Or maybe they've got another surprise

for us next week. For what it's worth, you know if you use any of those Adobe products. So it might be prudent to check in with them for any updates.

We've talked, we've touched a couple times recently on the VoIP DDoS attacks. They are continuing, like right now. A U.K.-based VoIP industry group representing the U.K. telecommunications sector said last week that several of its members active in the VoIP market have been hit by Distributed Denial of Service attacks over the past month. In a statement on Tuesday, the Comms Council of U.K. said the DDoS attacks were "part of a coordinated extortion-focused international campaign by professional cybercriminals." And what unfortunately we're beginning to see here is sort of another trend of malicious conduct on the Internet.

The organization did not share the name of the victims explicitly, but VoIP providers including 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 by the end of September.

The threat actors first launched DDoS attacks, then sent email demanding huge payoffs to stop the attacks. They knew that companies such as VoIP providers could not afford to remain offline without incurring huge financial losses and getting a lot of pressure from their own customers, whose VoIP was down and gone. So it had secondary effects. 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. We don't have ready tools for filtering VoIP traffic.

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." And Cloudflare, as we know, 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, unlike what was true in the case of the ransomware guys, these VoIP 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 their 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 U.K., said the attacks had extensive secondary impacts upon "critical infrastructure organizations including police, NHS, and other public services." He described the DDoS extortion campaign as "attacks on the foundations of U.K. infrastructure." And, you know, I guess everybody would like to escalate this to terrorism; right? Because as soon as you do that, then you've got different things you can do legally as we discussed last week, and as did happen when U.S. law enforcement went after the REvil guys.

So coordinated DDoS attacks against selected industry sectors have occurred in the past, and they appear to focus on industries that cannot afford to go offline even briefly. We talked about the original DDoS attacks against gambling sites which were desperate to remain online during major sporting events. So like, you know, the attacker would blast them offline during one painful period, and then say, okay, look what we can do. You want more of that? If not, pay us. And some did.

And similar attacks occurred a year ago, last September 2020, when attackers launched a campaign like that against EU-based Internet service providers. Other campaigns have targeted entities in the financial sector, banks and stock markets. Another recent sector

being targeted with DDoS extortion campaigns has been privacy and security-focused email providers. I think ProtonMail was being blasted not long ago. Also Runbox, Posteo, Fastmail, TheXYZ, Guerilla Mail, Mailfence, and Kolab Now. Oh, and also RiseUp. All of them email providers. All of them have been suffering attacks.

So what's happening is we know that the Internet has many massive botnets which are capable of producing astonishing levels of aggregate bandwidth attack. And the lack of egress filtering allows their outbound packets to carry any IP address they choose. In other words, they're able to spoof the source IPs, making it much more difficult to track them down. If there was egress filtering in place, they would at least have to spoof within that net, and it would be possible to block the net that they're in.

A long time ago this podcast looked at the incredibly elegant simplicity of the autonomous packet routing invention, you know, which is the Internet. Unfortunately, for all its elegant simplicity, it was never designed to prevent its own abuse. It really can't do so. And so today we have ever more attack targets hiding behind the services of Cloudflare and other attack mitigation services. They have no choice. If you're a target, and you're being blasted off the Internet, you need protection. The Internet itself can't provide it.

I got a kick out of this. Talking to Leo last week about "Dune," which we both loved, and it's come under universal acclaim, I mentioned how, like, I couldn't understand that these guys were using swords rather than handheld laser guns. It's like, or beam weapons. Like it's supposed to be the year, what is it, 10191? And now I confess, Jason, I'm not a "Dune" guy. I read the first book. I get it that there's lots more and all that. Anyway, I got a tweet from David N., whose Twitter handle is @RandomDaveInFL. He explained this.

"Hello, Steve. Just listened to the latest podcast." He says, "I'm a bit late in listening this week. And you probably already know about 'lasguns'" - okay, that's a thing, that's the official term, L-A-S-G-U-N - "lasguns in Dune." And so I didn't know, so thank you, Dave. "But there actually is an answer why they don't use them more often. Turns out the interaction with the lasgun and shield" - that is, you know, the personal shields they all wear - "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 until Part 2. Only wish it was sooner. Have a great day."

And he gave me a link in his tweet to a page, dune.fandom.com. There's a wiki there, and there's a page on the lasgun where, I'm just excerpting from it, it explains: "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." And it goes on: "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."

So all you "Dune" people out there, you probably knew that. I didn't. So I wanted to share it with the rest of us who also didn't know. Makes sense. And I do like when sci-fi imposes restrictions of that kind that you then need to work around, and they can often give you interesting plot vehicles that you wouldn't otherwise have. Because if you can just blast people from a distance, that's not very sporting.

JASON: Yeah. Not everybody has to love the laser, as much as all sci-fi seems to focus on. I haven't seen the movie yet. I don't know why. But I need to see it. I mean, literally everything that I ever hear anyone talk about it is like extremely positive. So I'm waiting for the right time to see it with my wife.

Steve: It was extremely positive. On the other hand, it was extremely half the story.

JASON: Yeah, I keep hearing that, too. It's like you want the rest.

Steve: And many people, I remember a friend of mine said he was looking at his watch, and it was like two hours and 15 minutes in, and he was thinking, how long is this movie? And they still have a lot of ground to cover.

JASON: Right.

Steve: And then it ended.

JASON: There's no way they're rounding this out.

Steve: Yeah. And then it was over. He's like, what? Wait, wait, what, huh? So anyway, I will watch it again in two years when Part 2 is available.

JASON: Yeah, yeah.

Steve: So that I can spend five hours of my day having the whole experience at once. But, I mean, it sounds to me like maybe you could wait until Parts 1 and 2 are both available.

JASON: Nah, I'm not going to be able to wait. I'm going to have to watch it.

Steve: Well, for what it's worth, HBO Max, which co-released it because they had the deal with Warner Bros., it's only till the end of this month, I think. I hope it wasn't the end of last month. But it's a limited release streaming. So if you want to not watch it in the theater, but get it from the comfort of your own couch, you can't do it in December.

JASON: We've got to do it soon.

Steve: So maybe it'll be somewhere else, or maybe it'll be Pay Per View or something. But, you know.

JASON: Right.

Steve: For anybody else who didn't know that, it's only available for a limited time.

Peter G. Chase, he said: "I'll be reading" - and this also refers to a grumbling that I had last week that I shared with Leo. He said: "I'll be reading a newsletter and get the overlay" - meaning over the screen - "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." We were talking about how I'm noticing that many pages are now apparently monitoring the user's mouse. And as you go to slide off the page up to the tab in order to close it, or maybe go to the back button, that seeing you about the leave, the screen will darken, and you'll get a, oh, before you leave, consider the following. And it's like, ugh. That is so annoying. And it's like, let me leave. Anyway, thank you, Peter.

JASON: Don't like that.

Steve: And JD Ainsworth, he said - oh, and he explained another puzzle from last week. "At Microsoft, PM is usually Project Manager or Program Manager, and S is either senior or Security." So Leo and I were puzzling over a term in a Microsoft posting, the SPM. You know, contact your SPM. It's like, what the heck? Anyway, he says: "SPM is most likely Security Program Manager." So JD, thanks for the clarification.

Last week, actually for several weeks, I had been talking about that everybody had the fourth prerelease of SpinRite, and I was working to get the fifth prerelease out to our testers. That happened Sunday afternoon, two days ago, at around 4:00 p.m. I posted SpinRite 6.1's fifth prerelease. 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 rewritten benchmarking system with the new feature I had just added. Once SpinRite has benchmarked a drive, which it's able to do very quickly, 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 that came out of this is how fast and practical 6.1 is turning out to be. I have some numbers. A 1TB Crucial SATA SSD, 1TB, can receive a SpinRite full Level 2 scan in 29.7 minutes. Okay. That's 1TB in half an hour. As I've mentioned previously, I was originally hoping for 0.5TB per hour from the new SpinRite. This is four times faster than that, which certainly sets a new land speed record for SpinRite. A 256GB SATA SSD was scanned in 7.3 minutes; a 129GB Kingston SATA SSD in 3.7 minutes. A spinning Seagate 6TB SATA drive took 9.27 hours.

Now, 9.27 hours, that's a while. On the other hand, it's not 9.27 months, the way it might have been. And that's 6TB. That's a big drive. You know, you don't format those anymore, right, like the way you used to, where you actually go out and look at the surface? No. You kind of can't format a big drive like that. So pretty much SpinRite is the only way that surface is ever going to get seen is by running a drive that size on SpinRite.

I also had some old IDEs. An old spinning 40GB Maxtor IDE with a parallel cable it was able to scan in 31.1 minutes. An old 160GB Seagate IDE, also parallel cable, took 48.1 minutes. So four times larger than that 40GB Maxtor, but a lot higher sector density, so a higher data rate. So four times the drive size in, as opposed to 31 minutes, 48 minutes. I had two drives on USB. A 64GB SanDisk USB-attached thumb drive took 59 minutes, so just shy of an hour. And a 1TB USB-attached solid state drive took 14, or is estimated to take 14.46 hours.

Now, yes, 14 hours for a terabyte on USB is way more than, what was it, 29 minutes for a terabyte on a SATA hooked to an AHCI controller. As I've said before, we won't have SpinRite's new super-speed performance for USB, which by the way will match the other drives in performance, until 7.1, 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 over to its new 32-bit OS home, so that it will be able to boot on UEFI-only systems, which no longer have a BIOS, that means they no longer allow to DOS to boot, and that's where SpinRite lives today. So I'm moving it over to the other OS. So it was overall much more efficient to wait to add USB support until we're over on the new platform so that I'm not having to rewrite everything there. But even 14.46 hours for a terabyte is significantly more practical than SpinRite has ever been before.

So anyway, aside from performing lots of fun benchmarks, although the fifth prerelease resolved a bunch of previous edge cases, it also revealed a few issues with specific hardware where I'll be spending some more time. But overall, things are looking very good. As a matter of fact, during the time before the podcast started, Jason, I was mulling over some of the feedback that I already had and wondering why really big drives were reporting an error that they shouldn't have reported.

And so I went into SpinRite source code before we began the podcast to at least answer for myself that question. And I realized I had a kind of a typo where I had written Int 13 ext version; or rather, where I wanted to have Int 13 extensions, I had Int 13 ext version. So I got the wrong variable name, which was testing a version number rather

than some flags, which completely explains a large class of the feedback that I got. So anyway, it's looking really good, and I'll fix those, and then we'll be moving forward again. So just a word of thanks to everybody who's testing it for us.

JASON: Right on. All right.

Steve: We'll take our last break, and then we're going to talk about Trojan Source Code.

JASON: That just rolls off the tongue right there. Trojan Source. What is Trojan Source?

Steve: So what's so cool about this is that everybody listening is going to be able to get this. It's kind of like one of those "Oh, my god," like where it hadn't occurred to you, but then when you get it, it's like, oh, this could be bad. And indeed, that's the other thing that is cool in a spooky way is, like, this could happen. And I'll save the punch line for the end.

But a pair of researchers in the U.K. have been exceedingly clever. They 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. I have a link at the end of the show notes to their 15-page research paper, but I'll just share the abstract. And I'm going to share some chunks of it because they do a good job of explaining this.

Their 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're displayed, leading to vulnerabilities that cannot be perceived directly by human code reviewers." In other words, you'd be like looking right at the source code, and what the compiler produces from what you're seeing is different from what you see.

They said: "'Trojan Source' attacks, as we call them, pose an immediate threat" - and I agree - "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."

Okay. So what exactly have these clever guys come up with? They introduce their ideas quite well, so I'm just going to share their introduction. They said: "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's well-known that malicious compilers can produce binaries containing vulnerabilities. As a result, there's 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," they say, "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, or contributing subtle vulnerabilities into open-source projects. In order to prevent or mitigate such attacks, it's 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."

They say: "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" - 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 use to disclose this vulnerability across the industry. And we raise a new question about what it means for a compiler to be trustworthy."

Okay. So how's this done? 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 that website. But in a homograph attack, the domain they're actually on looks like PayPal, but it isn't.

Okay. The way these guys formally state this 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" - there's that word, an interpreter, remember, danger danger - "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."

Okay. Now, not all languages, as I'm sure everyone knows, 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, which is to say 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. An example is RLE, that is, the single character has the abbreviation RLE, which stands for right-to-left embedding. RLO stands for right-to-left override, for example.

And since these are locally modal, that is, it's like an escape character that sets the direction, how do you unset it? It's necessary to have some way of undoing them. So there's a, for example, PDF character which stands for Pop Directional Formatting, which literally is like popping the stack. It restores the previous direction. And in the definition it states for PDF states "terminates nearest LRE, RLE, LRO, or RLO." So anyway, it means that Unicode rendering is an interpreter which is maintaining some 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," they write, "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."

Then in their paper they have a table of the nine different things. And they say: "The table provides a list of Bidi override characters relevant to the 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." Which is, again, that pop. Since our listeners cannot see the table, LRI stands for left-to-right isolate is the term they use, RLI for right-to-left isolate, and PDI for pop directional isolate. Anyway, we'll get to "isolates" in a minute.

So they said: "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." So then they have the Unicode character RLI, then a b c, then PDI, which pops that RLI effect. And they say: "...which will be displayed as." So again it was RLI a b c PDI will be displayed as c b a. They said: "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," or the proper number of pops.

Okay. So the ability to reorder individual character sequences, to that we add one additional complexity which adds the ability to use tricky Unicode encodings, and that's those isolates I talked about before. In the Bidi specification, isolates, sort of the way their name sounds, are groups of characters, you know, isolated, that are treated as a single entity, so grouped together. Okay. That is, the entire isolate will be moved as a block when the display order is overridden. And moreover, isolates can be nested.

Okay. So here's an example. We have RLI, which starts a right-to-left isolate. Then we have LRI, which starts a left-to-right isolate. Then the letters "abc." And then we pop the preceding RLI with a PDI character. Now we start another isolate with an LRI, and we have "def." And then we close that by popping it with a PDI. And then we close that original RLI, which encloses the two internal LRIs with a final PDI. So what was written there is "abc," and then "def," with all these various control characters bracketing them. What will display is "defabc." In other words, what you see is none of the control characters. You see the def block preceding the abc block. But the compiler, which ignores the Unicode control characters, sees the abc block followed by the def block. In other words, using this trick it is possible to swap blocks of characters, and all compilers and languages currently fall to this.

Okay. So by this point everyone gets it. 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," they write, "to create adversarially encoded text that is understood differently by human reviewers than by compilers." And they did it. They

pulled it off. They successfully implemented invisible malicious changes to the source code of all those languages. In their paper they take each language at a time and give examples of how these techniques can be used in the real world.

So we have a new, 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 us, and of course all the bad guys throughout the world. They wrote: "We contacted 19 independent companies and organizations in a coordinated disclosure effort to build defenses for affected compilers, interpreters, code editors, and code repository frontends. 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, maintain their own internal compilers and build tools. The initial disclosures were sent on July 25th, 2021. Several of the initial recipients asked us to include additional organizations in the disclosure process, and we did so. We also sent additional disclosure throughout the embargo window for affected products that we discovered during the disclosure process.

"Of the 19 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." Okay, nine out of 19. "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." Doesn't seem like very much money for this research. This is serious.

"On September 9th, 2021 we sent a vulnerability report to 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, opted in to the VINCE tool for these shared communications. CERT also added three additional vendors to the disclosure beyond the 19 we had already contacted." On October 18th, Trojan Source attacks were issued two CVEs for tracking the Bidi attack and for tracking the homoglyph attack. They noted that it's also possible to fool that way. 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, which contains representatives of the security teams of 21 operating systems as of the time of this writing. The list coordinates the application of patches across OS maintainers, but allows a maximum embargo period of 14 days."

Okay. And finally, this. "We were curious," they wrote, "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 the 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 '21.

"The majority of the results were false positives. Examples of clearly non-malicious encodings included LRE characters placed at the start of the file paths" - you know, left-to-right encoding, and they already were left-to-right encoded - "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 the 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, dumping the passwords file. In other words, these techniques had already occurred to someone and had already been under exploitation in the field. Not widely, but then neither was their scan that wide. They also have some very good thoughts about the implementation of defenses for this class of attacks.

They said: "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."

And they go on. Basically they identified something that occurred to them as something nobody had ever thought of. They fully implemented exploits in a handful of languages. They quietly told everybody who might be affected that, hey, pay attention. This is real. This is bad. Half the people understood; the other half are idiots who said, you know, call our attorney. And good luck to them because I guarantee you this is going to, you know, bad guys are going to jump on this, now that this is public, and look for places where this can be leveraged. I really, really hope that the industry had time to bring up its defenses and prepare for this because this is going to slip under an awful lot of radar. Anyway, Trojan Source. I thought it was very cool.

JASON: That's fascinating. Yeah, that's some pretty intense stuff there.

Steve: Yeah.

JASON: Right on. Well, excellent stuff, excellent work, Steve. Always appreciate your deep dives on this stuff. I will admit there were a lot of letters flying by there for a second. If I wasn't looking at what you were reading, it was really hard to follow along. But that's a complicated thing, and appreciate you breaking it down for us, as you do each and every week.

Steve is awesome. If you want to check out everything that Steve's up to, go to GRC.com. All of Steve's goodness, from SpinRite, of course, to SQRL. Information about SQRL can be found there. You can find audio and video of this show there, as well as transcripts. It's the only place where you can find transcripts for Security Now!, even.

And then of course our website, TWiT.tv/sn for Security Now!. And you can find the audio and video there, as well as all the links that you need to subscribe to the feeds, even on YouTube. You can jump out to YouTube and do it there. It's all listed. And we do record this show live every Tuesday. So if you want to tune in to that, that's not a problem, as well. TWiT.tv/live, 1:30 p.m. Pacific, 4:30 p.m. Eastern, 20:30 UTC. I had to look that up, so I think it's right, at least for one more week. Things are going to change next weekend. So we'll have to prepare Leo for that.

But speaking of Leo, Leo will be back in the hot seat next week with you, Steve. So everybody can look forward to having him back then, and look forward to seeing you, too, as we do each and every week on Security Now!. And thank you for allowing me to crash the party for a week again.

Steve: Oh. Allowing you? Thank you for keeping it going. Great to see you, Jason.

JASON: Thank you, Steve. Thanks to everybody. Great to see you, too. We'll see you next time on Security Now!. Bye, everybody.

Steve: Bye.