Use After Free

Description: This week we look at the U.S.'s new cybercrime reporting law that was just passed. We examine a worrisome software supply chain sabotage and the trend it represents. We look at "Browser-in-the-Browser," a new way to spoof sign-in dialogs to capture authentication credentials, and we examine the way MikroTik routers are being used by the Trickbot botnet to obscure their command and control servers. A very concerning infinite loop bug has been uncovered in OpenSSL - time to update! - and CISA walks us through their forensic analysis of a Russian attack on an NGO. We then take a look at the Windows vulnerability that refuses to be resolved, and we'll finish by spending a bit more time than we have so far looking more closely at why Use After Free flaws continue to be so challenging.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-863.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-863-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. There's a new sheriff in town, a new U.S. law requiring cybercrime reporting. We'll take a look at that. A supply chain sabotage from one of the developers and maintainers of Node.js. And a new way to spoof sign-in dialogs, it's called "Browser in Browser," and it's scary. Plus Steve will take a deep dive into a common source of security flaws called the Use After Free bug. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 863, recorded Tuesday, March 22nd, 2022: Use After Free.

It's time for Security Now!, the show where we cover your security, your privacy, everything you need to know about staying safe online with this cat right here, Steve Gibson, GRC. He's doing the - now I know why you love Bobiverse, by the way. It's Star Trek-focused. There's a lot of Star Trek jokes in there.

Steve Gibson: There is actually. In fact, you won't know it - I don't remember if it's in the third book or the fourth, but there's actually - there's this thing that he calls, how would I say it, replicative drift, where...

Leo: Yes, because even though he's replicating himself, each one is slightly different.

Steve: Which, you know, it's like why is a computer going to have that, but it's a required, you know...

Leo: It's an AI. It's an AI.

Steve: Well, but...

Leo: And you need it, that's right. Otherwise they'd all be the same.

Steve: Exactly. So it's a required plot vehicle for him in order to get different Bobs, as all these Bobs keep replicating. So what happens is over time there are some - and of course you're right, Star Trek is a theme through the series.

Leo: Well, one of the Bobs names himself Riker.

Steve: Yes, Will.

Leo: Bob Number Two, yeah.

Steve: Yes, in fact, and he later drops Riker and just switches to Will.

Leo: Will, that's funny.

Steve: But still it's that connection. But then what happens is, as things go on, they start involving themselves with other cultures. I mean, not in a bad way. Like to help out these other intelligent sentient aliens who they find.

Leo: They violate the Prime Directive?

Steve: Yes. And there is a group that forms that are all up in arms about this, and they literally get labeled "Star Fleet."

Leo: I'm just finishing the first book because you shamed me last week, saying have I read it yet, and I hadn't started it. And I love it. It reminds me so much of "The Martian" and "Project Hail Mary" in the attitude of the narrator. Of course I'm listening to Ray Porter's narration, it's the same guy. So he's got that same kind of funny attitude. And in the science writing and the problem solving, it is another Andy Weir. Andy Weir 2. It's really good. It's called "The Bobiverse." First one's called "We Are Legion," by - I can't remember the guy's name. Something E. I know his middle name is E.

Steve: Well, in fact I was just thinking about this yesterday. There's a - you know, there's no spoilers here. But there's a moment where there's a problem that's being solved, and they're needing to rotate their passwords and update their keys. And I'm like, I mean, that's what the guy is talking about.

Leo: I know, I love it.

Steve: No wonder our listeners love this stuff, yeah.

Leo: Dennis E. Taylor.

Steve: Dennis Taylor.

Leo: And they're great, I have to say. And I've only read the first one. But so far I'm really, really enjoying it.

Steve: Yeah, you won't be let down. So today we are at 3/22/22. And there wasn't any big, I mean, there's like a lot going on, of course, because we still have all of this tension globally. But a number of our listeners have written with questions about Use After Free. We've talked about this vulnerability, how it continues to keep being a problem. And when I saw that, after looking at everything that I wanted to talk about this week there wasn't anything that was clearly topical, I thought, well, I'm just going to spend some time digging a little bit deeper into this issue of memory management in our contemporary computer systems to answer sort of as a group a lot of these questions that we've had. So I titled today's podcast "Use After Free," just because it's the way we're going to wrap things up.

But first we're going to take a look at the U.S.'s new cybercrime reporting law which finally did get passed and signed into law last week. We're going to examine the most tweeted to me thing of the week, which was a very worrisome software supply chain sabotage. And even more worrisome than one instance is that it turns out this is looking like a trend. And it is concerning.

We're going to look at the browser-in-the-browser exploit, a new way to spoof sign-in dialogs to capture authentication credentials; and examine the way MikroTik, or as I originally called them "Microtic," routers are being used by the Trickbot botnet to obscure their command-and-control servers. We've got an infinite loop bug of great concern which has been uncovered by our friend Tavis Ormandy at Google in OpenSSL, which is what makes it a big concern - time to update - and CISA walks us through their forensic analysis of a Russian state-sponsored attack on an NGO, a non-governmental organization. We then take a look at the Windows vulnerability that refuses to be resolved. And we'll finish, as I said, by spending a bit more time than we have so far looking more closely at why Use After Free flaws continue to be so challenging. And we also, for those who are coders, we have a terrific Picture of the Week.

Leo: It's kind of relevant, too. Hey, there's also a news story that I want to point you to and let you chew over it before we talk again. Of course you spent many years working on SQRL, a better way of doing passwords. I don't know if you saw this, but the FIDO Alliance has said, oh, we think we now know how to enter a passwordless world. They say, I'll just quote a little bit of it: "The passwordless FIDO standard already relies on device biometric scanners to authenticate you locally without any of your data traveling over the Internet to a web server for validation.

"The main concept FIDO believes will ultimately solve the new device issue," because of course switching and adding devices becomes the problem, "is a FIDO credential manager," this sounds a little bit familiar, "which is somewhat similar to a built-in

password manager. Instead of literally storing passwords, this mechanism will store cryptographic keys that can sync between devices and are guarded by your device's biometric or pass code lock." I think yours is a little better, to be honest.

Steve: Yeah, mine only needs one, and it synthesizes them on the fly based on where you are, as we know.

Leo: The problem is FIDO is Apple and Microsoft and Google, and you're just you.

Steve: Yup. And you have not seen me any spend any more time. After I spent seven years solving the problem, I dropped it like a hot rock and switched to SpinRite because I solved the problem. If the world wants it, great. Otherwise, fine.

Leo: Somebody should write to FIDO and say it's been solved, take a look. Because you address the exact thing they're talking about, the portability of these keys, and you address it in I think a better way than they're talking about.

Steve: Yeah. And, you know, all kinds of little edge cases like users telling the website, oh, I lost my key, please let me use something else. Well, that's an obvious exploit for spoofing. And so with SQRL you can set a little checkbox that says please don't ever listen to those requests. And so, I mean, and if you lose your credentials, how you can get them back, how you're able to obsolete lost credentials, I mean, I just - I did solve it once. And there it is. Meanwhile, I'm having a lot of fun making great progress with SpinRite.

Leo: Good. We'll hear more about that in just a second, and get our Picture of the Week, as well. Steve Gibson, Security Now!.

Steve: So this is just a fun one. It depicts the Six Stages of Debugging, which, as I said, most of us who have written code will understand. The first stage is the "That can't happen" stage. We would call that denial.

Leo: Denial, yeah. Not in my house.

Steve: Then we have, exactly, then the second one is "That doesn't happen on my machine." It's like, wait a minute, you know, it's something wrong with your computer. It's all good here.

Leo: I can't duplicate that, yeah.

Steve: That's right, exactly, can't duplicate it. Then the next stage, as the coder moves through this painful process, is "That shouldn't happen." And so Stage 3. Then when the coder has recognized, okay, well, I guess is it happening, we get to this fourth stage, "Why does that happen?" And then when that gets realized and internalized, we get to the fifth stage of, "Oh, I see."

Leo: In big letters. OH, I SEE.

Steve: Exactly. And then I love Stage 6, which is like I've felt it before. It's like, "How did that ever work?"

Leo: So true. So true.

Steve: Like, my god, how did it take until now to find that? But, oh, yeah, that's the nature of the process. Okay. So last Tuesday a $1.5 trillion government funding bill was signed into law. I think that keeps us ticking until September. President Biden signed it. And we're talking about that because part of that, part of what was packaged into that legislation is something we've touched on a few times when it had periodically been discussed. But what was under consideration until now, and it was only under consideration, became law. That new law mandates that owners and operators of critical U.S. infrastructure, to be defined in a minute, must report - must report - when their organization has been hacked, and if a ransomware payment has been made.

So this so-called - it's called the "Strengthening American Cybersecurity Act" of 2022, and unfortunately no good acronym jumps out. SACA, I don't know. It's not very catchy. Anyway, it was attached to the spending bill, which requires, as I said, the critical infrastructure operators alert the Homeland Security Department's CISA, the Cybersecurity and Infrastructure Security Agency, within 72 hours, three days, of a breach; and within one day, 24 hours, if the organization makes a ransomware payment. This new legislation also grants CISA subpoena power, allowing it to compel testimony from any entities which they believe have not reported a cyber incident or a ransomware payment.

And what's interesting is that this move is a reversal in recent policy because this legislation was removed, or as the legislators like to say, "stripped out of" the recent annual defense policy bill where it first tried to make it into law. And that was signed a few months ago. So, you know, given this change of heart, one has to wonder whether the potential for attacks against U.S. infrastructure, which would be secondary to the Ukraine/Russian conflict, might be behind this change of heart. And my eyebrows rose when I was learning that this so-called "Strengthening American Cybersecurity Act" passed the Senate unanimously. So no dissent at all.

Leo: Wow.

Steve: Yes, let's do - I know. Like when was the last time something was unanimous in our U.S. Senate recently? So again, it's no skin off anyone's teeth. They're basically saying, you know, just notify us if something happens. So now CISA will have up to two years to publish a notice in the Federal Register on proposed rulemaking to implement the reporting effort. You know, nothing ever happens quickly. However, reporting on this has suggested that CISA might choose to move faster, thanks to the current heightened threat climate.

The Senate's Homeland Security Committee Chair Gary Peters, who authored and championed the legislation along with Senator Rob Portman, issued a combined statement saying: "This historic new law will make major updates to our cybersecurity policy to ensure that, for the first time ever, every single critical infrastructure owner and operator in America is reporting cyberattacks and ransomware payments to the federal government."

And additionally, Rob Portman, the panel's top Republican, said the legislation will "give the National Cyber Director, CISA, and other appropriate agencies broad visibility into the cyberattacks taking place across our nation on a daily basis to enable a whole-of-government response, mitigation, and warning to critical infrastructure and others of ongoing imminent attacks."

So as I was researching this I got to wondering, because it sort of seemed like a loophole to me maybe, what exactly constitutes critical U.S. infrastructure? We hear the term all the time. But who exactly would be required to report? So to answer the question, I attempted to read, and that was a mistake, some of the actual legislation. I don't recommend it because now I have even less idea how our government actually manages to work, and more amazement that for the most part it kind of seems to.

I did manage to track down Presidential Policy Directive 21, Section 2242 - that's right, Section 2242 - subsection 'b'. And under the heading there, "Designated Critical Infrastructure Sectors and Sector-Specific Agencies," it explains: "This directive identifies 16 critical infrastructure sectors and designates associated Federal SSAs." Those are Sector-Specific Agencies of which DHS is almost always their reporting-to agency, the responsible agency. Like DOE has the Department of Energy, DOD has the Department of Defense obviously, and so forth. But in general, DHS, the umbrella.

They said: "In some cases co-SSAs are designated where those departments share the roles and responsibilities of the SSA. The Secretary of Homeland Security shall periodically evaluate the need for and approve changes to critical infrastructure sectors and shall consult with the Assistant to the President for Homeland Security and Counterterrorism before changing a critical infrastructure sector or a designated SSA for that sector. The sectors and SSAs are as follows." And as I said, this was after like a lot of distillation and whittling it down because, oh, my lord.

Anyway, those 16 are: Chemical; Commercial Facilities; Communications; Critical Manufacturing; Dams; Defense Industrial Base; Emergency Services; Energy; Financial Services; Food and Agriculture; Government Facilities; Healthcare and Public Health; Information Technology; Nuclear Reactors, Materials, and Waste; Transportation Systems; Water and Wastewater Systems.

So anyway, I was curious. And in case anybody else is, those are the 16. So the bottom line is reporting attacks and extortion payments is no longer optional. It will be the law. Or it is now, although it looks like CISA has some - has to get like, since they are the reporting-to agency, they've got to figure out how to turn it into a sufficiently large bureaucracy. Okay.

Winning this week's prize for the most listener-reported incident was a very worrisome software supply-chain event. I don't think it qualifies for being called an "attack" since it was an inside job. Consequently most of the tech press is using the term "sabotage." NPM is the package manager we've discussed from time to time for today's number one most popular, most used, and most in demand programming language, JavaScript. NPM is the default package manager for JavaScript's super-popular runtime environment Node.js. And a very prominent Node module known as Node-IPC, as in interprocess communication, which is commonly used for local and remote interprocess communication which has support for Linux, macOS, and Windows is the subject here. Node-IPC has over 1.1 million downloads per week, which puts it up at the high end of popularity.

In the context of Java's, now, not JavaScript, Java's Log4j incident and vulnerability, we were talking about code dependency trees, the idea being that one software library relies upon several others, and they may rely upon others, et cetera, creating this rapidly

branching tree which has the effect of creating a deep set of interdependencies. Liran Tal at "snyk" - I don't know how you pronounce S-N-Y-K.

Leo: Snike?

Steve: Snike? Yeah. Maybe. I hope it's not Snike. That'd be bad. Where do you work? I work at Snike.

Leo: Snike.

Steve: Like, what, on purpose? Deliberately? Okay, anyway, he was the first to report their discovery of this problem. Liran wrote, he said: "On March 15, 2022, users of the popular Vue.js frontend JavaScript framework started experiencing what can only be described as a supply chain attack impacting the npm ecosystem. This was the result of the nested dependencies Node-IPC and 'peacenotwar' being sabotaged as an act of protest by the maintainer of the Node-IPC package."

He said: "This security incident involves destructive acts of corrupting files on disk by one maintainer and their attempts to hide and restate that deliberate sabotage in different forms. While this is an attack with protest-driven motivations, it highlights a larger issue facing the software supply chain, that the transitive dependencies in your code can have a huge impact on your security."

Okay. So the story started back on March 8th, at which time npm maintainer for this package, "RIAEvangelist" is his handle, his real-world name is Brandon Nozaki Miller, he wrote some code and published an npm package named "peacenotwar," presumably referring to what's going on over Ukraine. And that package describes itself: "This code serves as a non-destructive example of why controlling your node modules is important. It also serves as a non-violent protest against Russia's aggression that threatens the world right now. This module will add a message of peace on your users' desktops, and it will only do it if it does not already exist, just to be polite."

Okay. Now, first of all, it's interesting that Brandon writes that he'll only place his antiwar message on desktops that have not already received the message "just to be polite" because what Brandon then does is anything but polite. His peacenotwar package sat around for a week with hardly any downloads. Then he adds this peacenotwar module as a dependency to one of his other very popular modules. Apparently he has 40 that he's maintaining. In this case, he added it to the dependency of Node-IPC, which caused it to get sucked down 1.1 million times per week.

Then apparently still getting even more amped up over the Russia/Ukraine conflict, and feeling that he has the power to be more proactive, Brandon added some deliberately Base64-obfuscated sabotage code to his Node-IPC package. This updated release added a function to check the IP address of the developers who were using Node-IPC in their own projects. When an IP address geolocated to either Russia or Belarus is found, the new version wiped the file contents from the machine, replacing them with a heart emoji.

Leo: Well, at least there's love involved.

Steve: There is obviously quite a lot that's wrong about this. But on top of it just being plain wrong to take advantage of one's implicitly trusted position as a package

maintainer, to have one's package do something that it was clearly not sanctioned to do, Brandon's targets were being very poorly chosen. If we're to believe what we're told about the war sentiments of the Russian population, it's the younger coders and developers who are more likely to be using Russian propaganda bypassing measures and to have a much more Western view of what's actually happening over there. And along comes some idiot in the West who wipes out a sympathetic developer's files only because of where they are coding.

And I should mention, it's not in the show notes, but the EFF just went ballistic over this. I mean, whoa. And sad as all that is, even more sadly, Brandon is not alone. The count of similarly recently altered software has risen to 21 incidents, separately having been identified. I noted as I looked through the list, it was in Russian, so I wasn't able to read the details, but I could see that the popular GUI toolkit QT was on the list, up near the top, actually; and another is es5-ext, which provides code for the ECMAScript 6 scripting language specification. A new dependency named postinstall.js, which the developer added on March 7th, checks to see if the user's computer has a Russian IP address, in which case the code broadcasts a "call for peace." Whatever that is.

So while not all of these changes are destructive, as Brandon's ended up evolving into, they are almost certainly not what those who are using them want or have every right to expect. Now, Brandon quickly removed his malicious changes after the very predictable outcry arose from other enraged developers. But, boy, you know, he has certainly not enhanced his credibility or reputation. And as I said, it does also point out what is probably a larger problem with the fact that we've ended up putting ourselves in a position where we are trusting in the goodwill and ongoing goodwill and the unfaltering behavior of lots of random developers whose software we're using, and we link it into a dependency tree. And our code grabs it, no matter what it is, assuming that it's going to stay what it was before. And of course we have no guarantee of that being the case. So, yikes.

As I start to describe this latest attack I have to conclude that the bad guys really do have an advantage. They're able to sit back and steeple their fingers with a faraway look in their eyes.

Leo: Bwa-ha-ha-ha-ha.

Steve: Exactly, while they endlessly cook up one sneaky way after another to bend the technology we've innocently given them to their dastardly purposes.

Today we have the latest attack concept that's been named Browser-in-the-Browser. It could have been done before, but it just occurred to a pentesting guy. It's a novel new form of phishing attack which was designed and documented by a pen tester who goes by the handle "mr.d0x." It's obvious once you see it, but it would not be obvious to a user who were to encounter it when they're expecting to see it. It's able to perfectly and convincingly simulate the pop-up browser window that we're all encountering more and more often with the spread of those third-party single sign-on options embedded on websites, you know, the Sign In With Google or Facebook or Apple or Microsoft, whatever.

I got two screenshots side by side which this guy produced, demonstrating that there's no way to tell them apart. With page rendering which allows JavaScript, CSS, and HTML to perfectly recreate the appearance of a legitimate "Sign In With" whomever pop-up dialog, an innocent user has no idea. And mr.d0x notes that one of the things that makes this attack so effective is that the spoofed dialog clearly and prominently displays the exactly correct domain name, the verification of which we've been drilling into our users

for years, saying "Make sure you're at the right site. Double-check the URL for any typos or lookalike characters. Don't be fooled."

Well, then along comes this perfect replica of a third-party sign-in dialog that accepts their innocently entered username and password. And because the JavaScript just printed a fake URL at the top of the dialog, which is where they're expecting...

Leo: Oh, I see.

Steve: You know...

Leo: It's easy, I mean, I understand it's easy to duplicate the login screen. That's just it's all open. It's JavaScript and HTML. I get it. So instead of where the browser puts the URL, you just put a graphic.

Steve: Yup.

Leo: That looks the same. And how would you know unless you clicked it. I guess if you clicked it, you might not be able to click the padlock, for instance, I don't know. Oh, that's really interesting. Wow.

Steve: Isn't it? I mean, and again...

Leo: Of course, yeah, easy.

Steve: You click on Sign In With Facebook. Up this thing pops. And you think, oh, I mean, you're expecting it. And so you may be doing due diligence. You go, okay, wait, F-A-C-E-B-O-O-K dot com, yup, that's Facebook. Nope.

Leo: Wow.

Steve: Oh, and Leo, just because mr.d0x - I couldn't remember if it was dr.d0x or not.

Leo: Dr.d0x would be better, but I guess he didn't get a degree.

Steve: Anyway, mr.d0x, yeah, to avoid any unnecessary duplication of effort, mr.d0x has thoughtfully provided attacker-ready, easy-to-use downloadable templates.

Leo: He even has dark mode.

Steve: That's right. You've got your dark mode, you've got your light mode, you've got your Chrome on Windows or macOS, whatever you need, just get it from GitHub.

Leo: Yeah. So you can't really just say look at the URL anymore.

Steve: Nope.

Leo: I do wonder, though, what would happen - I guess you could write some JavaScript that if you clicked the padlock you've get something else.

Steve: Probably another little thing that says, yeah, it's all good.

Leo: Oh, yeah.

Steve: Yeah, because the attacker has control of the window, so it can do whatever you want. I mean, it's a fake URL, so yeah, fake the click on the padlock. Yeah. And actually you'll be, since we were talking about SQRL, this already occurred to us back then. And a lot of attention was given to designing a pop-up dialog that could not be spoofed in this fashion. So again, we did a lot of work at the time. But yeah, wow. I just, you know, look what we've done to poor users.

Leo: I'm sorry. I'm sorry.

Steve: You know? No wonder your mom says, "Leo, how do I, dot dot dot."

Leo: What's going on? By the way...

Steve: And of course...

Leo: I know you're not reading the Bobiverse, you're not listening to it as an audiobook. You're reading it. But Ray Porter reading it does all the voices. He does Homer. Not a great Homer, but he does it.

Steve: Oh, no kidding.

Leo: So he's doing a lot of good voices in there. It's pretty funny. He does - Colonel Butterworth he does perfectly. It's very funny. Anyway, sorry, didn't mean to distract. Yeah, this is crazy. What do we tell people now? We always said make sure it's the right URL. You have to manually enter it? But that's not going to work for an authentication window.

Steve: And, well, look at the stuff that follows it. It's like I used to joke, one of my best friends was a lot older than I. He has since passed. But his name was Gary. I used to joke that the Windows, what is that key we type in, the five groups of five, I forgot what you call that. Anyway, it was...

Leo: The serial number, you mean, yeah, yeah.

Steve: Yeah.

Leo: And you had to type it in.

Steve: Yes.

Leo: The license key, yeah.

Steve: The license key. I used to joke that it was the equivalent of copyright protection...

Leo: Oh, it's terrible.

Steve: ...because he was never able to get it right.

Leo: Remember? We just typed that in. That was crazy.

Steve: Oh, yeah. He'd be looking at it going, okay, JQ5PP, yeah. And I'd go, Gary, no, that's an 8.

Leo: Oh, god.

Steve: But anyway, look at URLs. URLs used to be - they were supposed to be meaningful. Now they've got these GUIDs in the that's just gibberish. So, yeah, you can't ask a user to type that. They'd never get logged in. They'd just go away. So, wow.

Okay. Last Wednesday, Microsoft blogged under the title: "Uncovering Trickbot's use of IoT devices in command-and-control infrastructure." Microsoft considers MikroTik consumer routers to be IoT devices. That's how they lump them. And I suppose since they lack a keyboard or display, and they are by definition connected to the Internet, they qualify as Internet of Things. So Microsoft offered an interesting forensic write-up which I'll share the best bit of. They begin with a bit of history.

They said: "Trickbot, a sophisticated trojan that has evolved significantly since its discovery in 2016, has continually expanded its capabilities and, even with disruption efforts and news of its infrastructure going offline, it has managed to remain one of the most persistent threats in recent years. The malware's modular nature has allowed it to be increasingly adaptable to different networks, environments, and devices. In addition, it has grown to include numerous plugins, access-as-a-service backdoors for other malware like Ryuk ransomware, and mining capabilities.

A significant part of its evolution also includes making its attacks and infrastructure more durable" - and that's where we're headed here - "against detection, including continuously improving its persistence capabilities, evading researchers and reverse

engineering, and finding new ways to maintain the stability of its command-and-control framework.

"This continuous evolution has seen Trickbot expand its reach from computers to Internet of Things devices such as routers, with the malware updating its command-and-control infrastructure to utilize MikroTik devices and modules. MikroTik routers are widely used around the world across different industries. By using MikroTik routers as proxy servers for its command-and-control servers and redirecting the traffic through nonstandard ports, Trickbot adds another persistence layer that helps malicious IPs evade detection by standard security systems." Meaning that a security system might be banning connections to a known malicious IP, so no problem. Let's have Trickbot bounce its traffic through some random MikroTik router that's not going to be banned, and then it will send its traffic on to the command-and-control server, thus serving as a proxy.

Okay, so when Microsoft says that MikroTik routers are being used as command-and-control proxies using non-standard ports, they mean that once they've managed to crawl inside - "they" the bad guys - inside a MikroTik router, and we'll discuss exactly how that happens in a minute, they set up a proxy server, or actually the equivalent, it's actually just persistent NAT, that's listening on one or more non-standard ports for incoming command-and-control queries from instances of Trickbot out in the field. This provides the attackers with much more isolation and resilience than if all instances of Trickbot were phoning home directly.

We've all seen the movies where tracing the traffic is thwarted by having it bouncing around among various intermediaries in widely geographically spread jurisdictions. And also, assuming that there's a sufficient available inventory of vulnerable MikroTik devices, and we'll talk about that in a minute, apparently that's not a problem, having different instances of deployed Trickbots connecting to widely dispersed MikroTik proxies makes finding the actual command-and-control servers much more difficult, and it makes the Trickbot botnet much more difficult to take down.

Okay. So Microsoft said: "The Microsoft Defender for IoT research team has recently discovered the exact method through which MikroTik devices are used in Trickbot's command-and-control infrastructure. In this blog, we'll share our analysis of the method and provide insights on how attackers gain access to MikroTik devices and use compromised IoT devices" - meaning these MikroTik boxes - "in Trickbot attacks. This analysis has enabled us to develop a forensic tool to identify Trickbot-related compromise and other suspicious indicators on MikroTik devices. We published this tool to help customers ensure these IoT devices are not susceptible to these attacks. We're also sharing recommended steps for detecting and remediating compromise if found, as well as general prevention steps to protect against future attacks."

And I'm not going to go into all that because it's sort of down in the weeds and only applies to people who have these routers. And if you've got a MikroTik router, just update it. Make sure it's running the latest firmware and that you're not using default passwords because there are three ways Microsoft has detected attackers gaining an initial foothold on these MikroTik routers. They will first attempt to gain access using default MikroTik passwords. It turns out that the old lesson has still not been learned, and that MikroTik's design wrongly provides for remote access under default credentials. Which is just hard to believe in this day and age.

As we know, default credentials, if any must exist, should never also be allowed to be used to enable remote authentication from the WAN interface. Any user wishing to enable WAN-side remote admin - maybe after first passing a sanity test. Why? Why do you want to do this? Are you really sure? It's really not a good idea. Anyway, should be required to create non-default credentials. Not doing this, not requiring this, is incredibly

poor security design in this day and age. Well, really ever. But now how do you excuse it?

Okay. So Microsoft said these bad guys will first try using the device default credentials. If that fails, then a brute force so-called "credential stuffing" attack is attempted. Microsoft wrote that they have seen attackers using unique passwords that were probably harvested from other possibly related MikroTik devices. Hey, it's worth a shot. Why not?

And then, believe it or not - and, oh, yeah, I know we all believe it - by exploiting CVE-2018-14847, a well-known and long since patched four-year-old vulnerability which will succeed on devices still running RouterOS versions older than 6.42, there's another way in. This vulnerability, which we covered back when it was news at the time four years ago, gives an attacker the ability to read arbitrary files from a remote MikroTik router, specifically a handy file such as user.dat, which contains the passwords.

And once in, they then change the router's password to retain control. Since any such router is probably long forgotten, that is to say a router that hasn't had its firmware updated in four years, is long forgotten, probably tucked away in some dark corner or a closet with a sign on the door which reads "Don't mess with anything in here," it's unlikely that anyone will be inconvenienced by having their router's password unilaterally changed by some remote hacker, let alone having it commandeered for use as a botnet command-and-control proxy.

The attackers then use the handy-dandy MikroTik command that redirects traffic between two external ports on the router, thus enabling and establishing a line of communication between Trickbot-infected systems and Trickbot's command-and-control, or perhaps another indirect link in the command-and-control chain, if they want to be more fancy like in the movies. MikroTik has a RouterOS containing very powerful SSH-accessible IP, firewall, and NAT stacks. It's this unique remotely accessible static NAT capability that makes MikroTik routers so useful as unattended communications proxies. Microsoft says that they monitored a Trickbot-infected device connecting to a MikroTik router over port 449, after which the MikroTik router connected to a Trickbot command-and-control server over port 80.

And as for today's inventory of MikroTik routers, a relatively recent report by Eclypsium published this last December found that "hundreds of thousands" of MikroTik routers are still vulnerable to remote exploitation despite patches having been available for them for years. And because these devices feature unusually powerful hardware, which is why they're so popular, they're seen as high-value targets.

Back in early December of last year, 2021, when covering this report from Eclypsium, our friends over at BleepingComputer reached out to MikroTik for some comment about the sad state of their otherwise very nice and quite powerful router's updating. MikroTik replied: "The Eclypsium report deals with the same old vulnerabilities we have mentioned in our previous security blogs."

Leo: Oh, exactly. Yes.

Steve: Yup.

Leo: This problem has been here since 2018.

Steve: "As far as we know," they said, "there are no new vulnerabilities in RouterOS. Furthermore, RouterOS has been recently independently audited by several third parties. They all arrived at the same conclusion. Unfortunately," they wrote, "publishing updated firmware does not immediately protect the affected routers. We don't have an illegal backdoor to change the user's password and check their firewall or their configuration. These steps must be taken by the users themselves.

"We try our best to reach out to all users of RouterOS and remind them to do software updates, use secure passwords, check their firewall to restrict remote access to unfamiliar parties, and look for unusual scripts. Unfortunately, many users have never been in contact with MikroTik and are not actively monitoring their devices. We cooperate with various institutions worldwide to look for other solutions, as well. Meanwhile," they finish, "we want to stress the importance of keeping your RouterOS installation up to date once again. That is the essential step to avoid all kinds of vulnerabilities."

Leo: Yes. True.

Steve: Yes. It is absolutely true. And I suppose we might be more sympathetic if they didn't also apparently have a default password that allows users to login remotely.

Leo: This has been covered before, again and again. Why are you bringing this up, Steve? We know that.

Steve: We don't know that it might have been changed in their recent later releases. But again, so what? What may have once started off as fiction in the movies is actually underway today. All of those hopping five times around the globe before finally connecting to the mothership, and it's like, oh, we can't - we don't have access into all those other geographies. So, oh, well. Yet there are hundreds of thousands of these things out there right now. Unbelievable.

Leo: Hundreds of thousands. Amazing.

Steve: Okay. This one's cool. You're going to like this one, Leo. A new and moderately severe problem has been discovered in all supported versions of OpenSSL. Now, it's officially being called a high-severity problem; but since problems in OpenSSL can be so much worse than an infinite loop, I think it's more likely of moderate severity. Nevertheless, especially in today's current climate with increased cyber hostilities, it does seem destined to be exploited because it is so powerful, so ubiquitous, and so easy to exploit.

The reason is that any Internet server or service running any Internet protocol whose security is being provided by OpenSSL, as most Unix, Linux, and a great many cloud-based services are, you know, OpenSSL is the default TLS connection provider for our Linuxes, for example, that any of those that has not just in the last week received update patches and been updated, as many Internet servers will not have been yet, which consumes a user-provided certificate such as a client certificate, can be taken down with a simple, deliberately crafted TLS handshake.

The trouble was discovered and reported by Google's Tavis Ormandy, who reported his findings to the OpenSSL team on the 24th of February, last month. The trouble Tavis found and worked out with David Benjamin, Chrome's TLS guy, is that a modular square

root function named "BN_mod_sqrt()," if provided with a maliciously crafted certificate to parse, will enter an infinite loop, thus pegging the processor at 100% and permanently hanging that thread. The CVE write-up of this problem is interesting. It explains.

It says: "The BN_mod_sqrt() function, which computes a modular square root, contains a bug that can cause it to loop forever for non-prime moduli. Internally this function is used when parsing certificates that contain elliptic curve public keys in compressed form or elliptic curve parameters with a base point encoded in compressed form. It is possible to trigger the infinite loop by crafting a certificate that has invalid explicit curve parameters. Since certificate parsing happens prior to verification of the certificate's signature, any process that parses an externally supplied certificate may thus be subject to a denial of service attack."

Okay. In other words, since the certificate's signature is checked for validity after the exploitable parameters are processed, anyone can trivially create a certificate that will completely lock up anything that's using OpenSSL. Their description continues: "The infinite loop can also be reached when parsing crafted private keys as they can contain explicit elliptic curve parameters. Thus the vulnerable situations include" - and this is from the CVE write-up - "TLS clients consuming server certificates, TLS servers consuming client certificates, hosting providers taking certificates or private keys from customers, certificate authorities parsing certificate requests from subscribers, anything else which parses ASN.1 elliptic curve parameters, and any other applications that use the BN_mod_sqrt() where the attacker can control the parameter values that are vulnerable to this DoS issue."

Okay. In the earliest OpenSSL 1.0.2 version, the public key is not parsed during the initial parsing of the certificate, which makes it slightly harder to trigger the infinite loop. However, any operation which requires the public key from the certificate will trigger the infinite loop. In particular, the attacker can use a self-signed certificate, for example, to trigger the loop during verification of the certificate signature. So you've got to sign the certificate rather than not care about the signature. Big deal. This issue affects OpenSSL versions 1.0.2, 1.1.1, and 3.0. It was addressed in the release of 1.1.1n and 3.0.2 on the 15th of March, so last week. That was exactly a week ago, last Tuesday. So it's fixed in OpenSSL 3.0.2, which fixed the affected versions 3.0.0 and 3.0.1, and was fixed in OpenSSL 1.1.1n, which affected everything, 1.1.1 up through 1.1.1m.

So again, note that OpenSSL 1.0.2, which is vulnerable, has reached end-of-life and is not being actively supported, so users are advised, if anybody still has 1.0.2, to upgrade to a new release branch as soon as possible because this one is not going to be fixed publicly.

And note also that many critical infrastructure-ish things routinely make use of certificate-based mutual authentication, where each of the two endpoints provides an identity-asserting and assuring certificate as a means for securely establishing their identities to each other. Not just IP addresses, but let's swap certs. And Leo, you and I are always talking about a cert being the right way to authenticate an SSH connection. So it's like that. Therefore any such system with listening ports exposed to the public Internet, despite its administrators being sure that it will only connect to other endpoints which provide the proper matching certificate or expected certificate, are immediately vulnerable to being taken down by anyone else on the public Internet.

Not surprisingly, this has all generated a great deal of interest, and GitHub has been buzzing. And all the work has been done, with someone else using the handle "catbro666" providing a working proof-of-concept certificate which will take down any unpatched instance of OpenSSL that makes the mistake of accepting any such certificate. The hosting processor is pinned, as I said, at 100% utilization, and nothing else gets done.

So far there are no confirmed indications of this thing being exploited, but for this one it's more a matter of when, not whether. There was one report of the Italian cert posting that this had been seen in exploitation. But my guess is that was a mistake since it was only a one-off instance of that being alerted. So probably nothing to worry about. But again, if you're using OpenSSL, and your system will accept a cert from somebody, you want to make sure that you update it. And this is only a week old.

Okay, this is really cool, CISA Alert AA22-074A. I want to share this interesting CISA alert which details a Russian state-supported attack upon an NGO, a non-governmental organization, because it contains some interesting and quite specific forensic detail from their investigation and may have some useful takeaways for some of our listeners. CISA's alert is titled "Russian State-Sponsored Cyber Actors Gain Network Access by Exploiting Default Multifactor Authentication Protocols and 'PrintNightmare' Vulnerability."

So CISA's report opens with a summary. They said: "The Federal Bureau of Investigation and Cybersecurity and Infrastructure Security Agency (CISA) are releasing this joint Cybersecurity Advisory to warn organizations that Russian state-supported cyber actors have gained network access through exploitation of default MFA protocols and a known vulnerability.

"As early as May 2021, Russian state-sponsored cyber actors took advantage of a misconfigured account set to default MFA protocols at a non-governmental organization, allowing them to enroll a new device for MFA and access the victim network. The actors then exploited a critical Windows Print Spooler vulnerability" - imagine that - "PrintNightmare to run arbitrary code with system privileges. Russian state-sponsored cyber actors successfully exploited the vulnerability while targeting an NGO using Cisco's Duo MFA, enabling access to cloud and email accounts for document exfiltration.

"This advisory provides observed tactics, techniques, and procedures, Indicators of Compromise (IOCs), and recommendations to protect against Russian state-sponsored malicious cyber activity. FBI and CISA urge all organizations to apply the recommendations in the Mitigations section of the advisory."

Okay. So here are some of the interesting details of this attack which were uncovered during its investigation. They said: "As early as May 2021, the FBI observed Russian state-sponsored cyber actors gain access to an NGO, exploit a flaw in default MFA protocols, and move laterally to the NGO's cloud environment." They said: "Russian state-supported cyber actors gained initial access to the victim organization via compromised credentials and enrolling a new device" - meaning computer, or I guess it could have been handheld - "in the organization's Duo MFA. The actors gained the credentials via brute-force password guessing attack, allowing them access to a victim account with a simple, predictable password." Okay. So they're calling it brute force, but apparently not much brute was needed, or force, for that matter, if it was a predictable password.

Okay. So now we're in. "The victim account had been un-enrolled from Duo due to a long period of inactivity, but it was not disabled in Active Directory. As Duo's default configuration settings allow for re-enrollment of a new device for dormant accounts, the actors were able to enroll a new device for this account, complete the authentication requirements, and obtain access to the victim network."

Okay, now, first of all, I'll just mention, because I don't mention it later, that's a bad default; right? But it's not the worst default, as we'll see in a minute. But this says that if an account is dormant and stops being supported, just using it again without needing to do anything else brings Duo's support for multifactor authentication on that dormant account back to life. Doesn't have to be that way, but it's the default.

They said: "Using the compromised account, Russian state-sponsored cyber actors performed privilege escalation via exploitation of the PrintNightmare vulnerability to obtain administrator privileges." And again, this is another reminder. It's easy to think of remote code execution attacks or remote code execution vulnerabilities as being really bad. But we keep seeing that the bad guys are able to get in as an unprivileged user. You do not want them to be able to elevate that privilege to admin. Anyway, PrintNightmare made it possible.

"The actors also modified a domain controller file." Get this: c:\windows\system32\drivers\etc\hosts. Anyone who's familiar with the Windows directory layout knows that that's where Windows hosts file lives, and it is still there and still effective to this day. And we may remember a little anecdote that I got caught out when the credit card processing service backend that we use changed their IP address, and I had stuck their IP address in GRC server's host file, and I couldn't figure out why isn't DNS working? Well, I overwrote it in the hosts file, as one can. Anyway, these guys modified a domain controller file, the domain controller's host file, thus redirecting Duo multifactor authentication calls to localhost instead of where it would normally go, the Duo server. This change prevented the multifactor authentication service from contacting its server to validate MFA login.

Leo: So it failed, of course; right? And locked you out.

Steve: It failed, of course. This effectively disabled multifactor authentication for active domain accounts because the default policy of Duo for Windows is to fail open.

Leo: To fail open, aghhhhh. I remember Bruce Schneier talking about how many enterprise tools fail open, including firewalls.

Steve: Leo, we wouldn't want to inconvenience anybody, especially not a Russian attacker.

Leo: Well, yes, exactly.

Steve: That would be an inconvenience.

Leo: Who wants to close them down? That's why I'm putting on my hoodie. I'd getting ready. Oh, wow.

Steve: I know.

Leo: This is a real problem because a lot of supposedly very secure places use Duo.

Steve: Yes, exactly. So, and then as if to apologize for mentioning this, CISA added: "Note: Fail open can happen to any MFA implementation and is not exclusive to Duo." Okay. But since the "tyranny of the default" is one of this podcast's core observations, I was curious about Cisco's default for their Duo MFA. And sure enough, in an FAQ over at Duo.com, in answer to the question, "How can I configure the fail mode?" Duo replies:

"By default, Duo Authentication for Windows Login will 'fail open' and permit the Windows logon to continue if it is unable to contact the Duo service. You can set the fail mode during installation to 'fail closed'..."

Leo: Oh.

Steve: Yes. During installation...

Leo: But it's not the default.

Steve: No, "...by deselecting the 'Bypass Duo authentication when offline' box during installation." Okay. But come on.

Leo: That's as if your two-factor authenticator, "You don't have the code? That's fine. Go on in." It's so dumb.

Steve: Yeah, exactly. And if an admin is installing this for the first time, and you don't yet know it all well, so you're going to leave all of the presumably recommended defaults alone, at least for the time being.

Leo: Sure, right, yeah.

Steve: We've all been there; right? So is there any reminder in the UI that that's the way it was set? No. Okay? But it's easy to change that later; right? No. Since they're answering the question, "How can I configure the fail mode?," meaning that presumably it's not obvious, thus needing to be answered in the product's FAQ, Duo then proceeds to explain: "To change the fail mode after installation, use the Registry Editor, regedit.exe..."

Leo: Oh. What? You can set it.

Steve: Yeah. We provide a key for that. Just dig down 12 layers in the registry.

Leo: What? Oh, my god.

Steve: ...with admin privileges, of course, to create or update, they say, the following registry value. Okay. Wow. So not really the "secure by default" operation that you'd expect from an add-on multifactor authentication system whose entire purpose is to meaningfully increase the system's security.

Leo: Yes.

Steve: After all, as you said, Leo, this system is now inconveniencing everyone all the time by having it, except the Russians, who simply bypassed it by taking strategic advantage of Duo's well-known default fail-open policy. So I wanted to bring this up in case any of our listeners work with companies using Cisco's Duo MFA.

Leo: I do, yeah.

Steve: It might be worth having your IT admin check out the current setting of that registry key. Anyway, I've included a link to the FAQ item in question in the show notes. Anyway, CISA continues: "After effectively disabling multifactor authentication, Russian state-sponsored cyber actors were able to successfully authenticate to the victim's virtual private network as a non-administrator user and make Remote Desktop Protocol connections to Windows domain controllers. The actors ran commands to obtain credentials for additional domain accounts, then used the method described in the previous paragraph, changed the MFA configuration file, and bypassed multifactor authentication for these newly created and compromised accounts.

"The actors leveraged mostly internal Windows utilities already present" - remember Living Off the Land - "within the victim network to perform this activity. Using these compromised accounts without MFA enforced, Russian state-sponsored cyber actors were able to move laterally to the victim's cloud storage and email accounts and access desired content."

And for those who aren't clear about the nature of NGOs, the target of this attack, they're typically benign, non-profit, explicitly unaffiliated with any particular government. A few examples of NGOs would be Greenpeace International, CARE, the World Wildlife Fund, and Doctors without Borders. CISA doesn't tell us which NGO was targeted in this instance. But Russia knows. Wow.

Then we have the Windows local privilege escalation, speaking of local privilege escalations, that Microsoft seems unable to fix. This is the, believe it or not, ongoing saga of a Windows flaw that Microsoft, paradoxically, appears to be unable to fix. Since there's no obvious reason why this particular problem should be beyond them, one must conclude, again, as we have been with increasing frequency lately, that they just really don't care, or that something has gone very wrong somewhere deep inside.

This all begins way back last August of 2021 with an elevation of privilege vulnerability in the Windows user profile service. Back then, it was tracked as CVE-2021-34484 after having been discovered by a security researcher named Abdelhamid Naceri. And if his name seems familiar it's because the problem he found just won't go away, and we've been following. And it's not a small problem. It's, as I said, a local privilege escalation which allows unprivileged users, like those pesky Russkies who manage to gain access, to then acquire valuable administrative privileges in Windows 10, 11, and the latest version of Windows Server. It has a CVSS of 7.8. And although exploits have been publicly disclosed, thus by Microsoft's own definition it's a zero-day for them, no evidence has yet been found of its exploitation in the wild or, for that matter, any sincere attempt on Microsoft's part to actually fix it. No discovery.

It was originally said to be fixed, as I noted, as part of August 2021's Patch Tuesday. But Naceri became curious after those patches were published and pushed out, to see how Microsoft had fixed what he had found. What he now found back then was that Microsoft had not fixed the problem. Instead, it was another one of those "address the symptom not the problem" pseudo-fixes. So Naceri presented another proof of concept which bypassed Microsoft's non-fix across all Windows desktop and server versions. And spoiler alert, that remains true today.

It was at this point that our friends over at 0patch, the guys who create those wonderful little itty-bitty micropatches which fix problems that Microsoft hasn't yet, or can't, or won't, decided to release one of their own unofficial patches for all Windows versions, making it free for all their registered users.

Apparently not being in any huge hurry to fix this after their August fail, the first time they checked this off their "fixed" list, Microsoft finally responded to this second bypass in their January of this year, 2022, Patch Tuesday. And they updated its CVE for some reason to 2022-21919, marking it again as fixed. And again, believe it or not, it wasn't, and today still isn't.

Once again, Naceri found a way to bypass Microsoft's second attempt, and noted that this one was even worse than the first. Happily, that original micropatch still worked. But March's, this just two weeks ago, update two weeks ago changed the DLL in question, which is prof as in profile, P-R-O-F-E-X-T dot dll, which broke 0patch's micropatch. So not only did Microsoft's March update still not resolve the problem, but it also removed the third-party protection that 0patch users had been enjoying until then. Undaunted, 0patch has updated its micropatch so that its users are once again protected while we presumably wait for Microsoft to yet again try to actually fix the problem.

And in one last bit of happy news, users of Windows 10 editions 1803, 1809, and 2004 have remained safely protected by 0patch's original patch since those Windows editions have reached the end of their support life and were therefore spared from receiving Microsoft's update, which reexposed everyone else to the problem. Yes, that's what makes the security industry and the computer industry so interesting [crosstalk] nonsense.

Leo: I can fix that. Again and again.

Steve: Wow, Leo.

Leo: All right. Let's talk about garbage collection and Use After Free, which sounds kind of sexy, actually. I don't know, what is Use After Free?

Steve: So today's nominal podcast topic was inspired by several pieces, as I mentioned at the top, of recent listener feedback. And it was a tweet from Stephen Bellchester that finally caused me to decide to give this a bit more time. Stephen tweeted: "It would be good if you could clarify your comments about use after free vulnerabilities in Firefox," he said, "(and similar ones in previous weeks)." He said: "You've said a few times that use after free issues are being caused by memory management being done automatically by the language. That implied, to me at least, that the language that Firefox is written in is memory managed and causing the issues, which seemed pretty unlikely to me. After some further digging it seems like the issue was actually with Firefox's automatic memory management that runs underneath the HTML/CSS/JavaScript stack."

Okay. So first of all, if I didn't say it, I would have meant that memory management is being performed by the language's runtime.

Leo: But you could have a Use After Free if you malloc something and deallocate it and then call it. You could do it to yourself. It doesn't have to be an automatic tool.

Steve: That is exactly right. And in fact I will get to that sort of non-automatic issue in a second. But the attacks that we see being leveraged are generally the implementation of JavaScript. And JavaScript is famously a language with automatic background garbage collection. And although unfortunately we have a name collision here, we made it very clear a long time ago that Java, the language, and JavaScript have nothing in common; right? But Java, the language, is a perfect example, both because being a virtual machine-based language, you know, there's the Java JVM, Java the language has a clear run-time environment, and because it's famous for having a very powerful garbage collector.

In looking around for some examples, I found a web page titled "How Java Garbage Collection Works." And it starts off saying: "Java Memory Management, with its built-in garbage collection, is one of the language's finest achievements. It allows developers to create new objects without worrying explicitly about memory allocation and deallocation, because the garbage collector automatically reclaims memory for reuse. This enables faster development with less boilerplate code, while eliminating memory leaks and other memory-related problems."

Then it says: "At least in theory. Ironically, Java garbage collection seems to work too well, creating and removing too many objects. Most memory-management issues are solved, but often at the cost of creating serious performance problems. Making garbage collection adaptable to all kinds of situations has led to a complex and hard-to-optimize system. To wrap your head around garbage collection, you need first to understand how memory management works in a Java Virtual Machine." And we're not going to get into that because the rest of that page goes into a lengthy treatise on Java's memory management. The task of sweeping out the memory garbage has been the topic of a great many academic papers.

And I found to my, I guess not surprise, but I thought it was interesting, that Amazon lists two books dedicated solely to the subject. There's "The Garbage Collection Handbook: The Art of Automatic Memory Management," and then it says "(Applied Algorithms and Data Structures)." And then we have "Garbage Collection: Algorithms for Automatic Dynamic Memory Management." And I thought the back flap of that one was interesting.

It says: "The memory storage requirements of complex programs are extremely difficult to manage correctly by hand. A single error" - like what you were talking about, Leo - "may lead to indeterminate and inexplicable program crashes."

Leo: Often does. Very common.

Steve: Worse still, and in fact that's what you want, actually, is a crash.

Leo: Yeah, you wish it would crash.

Steve: Exactly. Then you can begin to track it down. And the flap says: "Worse still, failures are often unrepeatable [oops] and may surface only long after the program has been delivered to the customer."

Leo: Yeah.

Steve: "The eradication of memory errors typically consumes a substantial amount of development time. And yet the answer is relatively easy, garbage collection, removing the clutter of memory management from module interfaces, which then frees the programmer to concentrate on the problem at hand rather than low-level bookkeeping details. For this reason, most modern object-oriented languages such as Smalltalk, Eiffel, Java, and Dylan" - and of course JavaScript - "are supported by garbage collection."

Leo: You can tell how old this book is by those choices of languages.

Steve: Yes. Dylan? Hello, Dylan.

Leo: Smalltalk and Eiffel? Okay.

Steve: Smalltalk, right.

Leo: They are garbage collected, that's true, yeah.

Steve: Yeah, in fact I think it was 1991, maybe.

Leo: The '90s, yeah, yeah, I think so.

Steve: '91, maybe, yeah. It says: "Garbage collecting libraries are even available for such uncooperative languages as C and C++. This book considers how dynamic memory can be recycled automatically to guarantee error-free memory management. There is an abundant, but disparate, literature on the subject, largely confined to research papers. This book sets out to pool this experience in a single accessible and unified framework. Visit this book's companion website" - which I'll be surprised if it's still around - "for updates, revisions, online GC resources, bibliography, and links to more GC sites." And speaking of the devil, Dr. Dobb's Journal reviewed the book.

Leo: Oh, yeah. Oh, yeah. "Running Light Without Overbyte." Oh, yeah.

Steve: That's right. They said: "Whatever else Java has accomplished, it has finally brought garbage collection into the mainstream. The efficiency and correctness of garbage collection algorithms is henceforth going to be of concern to hundreds of thousands of programmers. Those who really care about this could do no better than to start with 'Garbage Collection: Algorithms for Automatic Dynamic Memory Management,' the sort of comprehensive engineering manual that is so rare in computing."

Leo: Just before we go too much farther, just so people know why this important, almost always when you're doing a computer, let's say you're creating an array, or you've got a database. You're going to allocate some memory to store the stuff in, operate on it, and then when you're done, deallocate it. If you don't deallocate it, eventually memory fills up, it's called a "memory leak," and you crash. So you've got to deallocate it. And there are really only three different ways to do it. One, the way

Steve does it, which is he keeps track of it, and when he's done with it, he deallocates it; right? You do it by hand because you're in assembly language.

Steve: Yeah.

Leo: There is garbage collection, which is an automated process and can be done well or poorly, that periodically on a timer goes to the system and says, are you using that? Do you need that? Okay, I'm going to let it go. There's also what Rust does, which I think some people think is more effective, just another kind of garbage collection, which is counting by reference. So it just counts up how many times something uses it, counts down. When it's not being used, it releases it. All three have problems. There's no perfect way to do this because if you deallocate something and then use it, that's a Use After Free bug. So it's not unique to garbage collection. I think people would argue letting programmers do it is even worse.

Steve: So, in fact, yes. I mentioned in my notes, I said it could certainly be that in a non-automatic language like C, one thread or process manually frees some memory that another thread or process later attempts to use. In that case, the culprit is clearly a coding error. Either the memory should not have been freed when it was, or the code attempting to still refer to it later should have known not to attempt to do so. Or, as in an instance we referred to recently, there was no actual error in the code at all. But there was a dangling pointer left around which pointed to some previously freed memory. The original code was not going to refer to that pointer, but a clever hacker figured out how to exploit it to their advantage.

Leo: Oh. So you could have written a perfect code.

Steve: Yes.

Leo: Released some memory. And then the bad guy modifies your code and attacks you. Oh, that's devious.

Steve: Yes. Yes. And so what we see is that the term "Use After Free" is a bit of a catchall, more than we'd like it to be. It's a bit like when Microsoft describes a vulnerability by saying that it allowed a security restriction to be bypassed. Like, what?

Leo: Huh? Well, we, you know...

Steve: Yeah, yeah, we're not really sure, but yeah. It seemed bad. So probably because it allowed a security restriction to be bypassed. But for what it's worth, this issue is a big deal. You know, it's been the sole topic of many computer science Ph.D. dissertations. There are people with "Dr." in their name because they wrote something, they invented some new twist on garbage collection. I mean, you know, in scrolling down through a lot of this stuff, Leo, I saw simple examples, right, where many languages allow you to use a block structure where you'll open curly braces and do a bunch of stuff and then close curly braces.

And the presumption is that block is self-contained. So if inside those curly braces you were to just say, you know, A equals something or other, then the presumption is when you close the curly brace, because you implicitly declared and first used the variable A within the curly brace, that leaving the curly brace, that A is so-called it's "out of scope."

Leo: Scope, yeah. Scope variables. Almost all modern languages do that.

Steve: Yes. Yeah. And again, super handy, super convenient. But you just need the language to know that, oh, we're no longer responsible for talking about A. And if you do, it's the programmer.

Leo: It's an error. Well, and the compiler or the interpreter's going to go, hey, A's undefined. What are you doing?

Steve: Yeah, we don't know what that is now.

Leo: Yeah. The reason I'm defensive of GC is because every language I use has garbage collection. It's hard to find a language these days except for C and Rust and Assembly.

Steve: Well, and I'm not sure where we are today, but there was a time when languages, like the operation program would suddenly stall for no apparent reason. It's because the garbage collector had kicked in.

Leo: That's the problem.

Steve: It was, like, busy. Yes.

Leo: Yeah, you get these periodic pauses while going through the - little robots going around looking, got any garbage for me?

Steve: It's that guy with the little white moustache and...

Leo: On a wheel, he's got one wheel.

Steve: And he's got a little broom, and he's like, you know, sweeping along.

Leo: Yeah. I love this subject. Keep going. This is good stuff. This is also in some ways the history of computing, really.

Steve: Yeah, you know, as our old-time listeners know, we had a lot of fun back in the early days of the podcast talking about how this all happened and the whole history of computing. So anyway, all I wanted was that it is one of the reasons that our listeners

are sometimes confused is that Use After Free is not just one thing. It doesn't have one cause. It doesn't have one environment. I've had people say, hey, static languages don't have the problem. Yes, they do. Dynamic languages don't have the problem. Yes, they do. Again, it's like saying, oh, you know, this vulnerability is a security bypass. Okay, well. And so Use After Free just sort of is more generic than we would like, but so it serves as a catchall for a large set of environments where a hacker was somehow able to reference memory in a way that wasn't intended. And so we just kind of call that Use After Free.

Leo: And don't.

Steve: Yeah. And more specific details would be nice, if we had them.

Leo: Yeah. I never even thought about that. But of course if a bad guy could write code, try to access memory that had been deallocated, your code could be perfect, but the bad guy was able to modify that code. That's bad.

Steve: Yes. And what can happen is that if you've deallocated some code, and then it's been reused, then the old pointer used to point to something in a different context. Now it points to its new contents, which the attacker may have just managed to load.

Leo: Right.

Steve: So software is hard.

Leo: I used to play with and love a language called Forth. It was all on the stack.

Steve: The write-only language.

Leo: Yes.

Steve: You cannot read that language.

Leo: But you didn't really, I guess you could, as I remember, allocate memory. But the idea was it's a stack-based language. Everything's on the stack. But you could still overrun the stack. So even there.

Steve: Sure. And, for example, in C, local variables are on the stack because our computer instructions provide a very nice stack pointer, actually it's a frame pointer reference because the stack might still be moving around.

Leo: It's this part of the stack here, yeah, yeah.

Steve: Exactly, exactly. And so when you exit from the routine, you so-call "pop the stack," which just disposes of all of those temporary references.

Leo: This stuff is fascinating. People who end up getting into programming often end up into writing their own interpreters, compilers, languages because it is so interesting.

Steve: And we've also seen the phenomenon, if you give somebody am object-oriented language, which inherently allows you to create meta language to describe the problem domain, you can end up spending all your time coming up with the coolest way to start getting ready to actually write some code.

Leo: That's kind of like Forth, too. That's why it's read-only, because you write your own vocabulary. So you're writing in a language you understand, but nobody else does. And absolutely that's the big thing of Lisp and Scheme, the languages I love, is writing a domain-specific language. And you're right, you could spend your entire day doing that and not solving the problem.

Steve: Getting really ready to actually start.

Leo: I am so set.

Steve: I've got this cool vocabulary.

Leo: Steve Gibson. So much fun. It's really great to have somebody who has roots in the past, understands the present, and can contextualize what's going on. That's what this guy does so well, every Tuesday, right here, 1:30 Pacific, 4:30 Eastern, 20:30 UTC. If you want to watch it live, live.twit.tv. People who are watching live often head over to our IRC, yes, we run an IRC server, well, our community does, at irc.twit.tv. You could chat live there. People who are members of Club TWiT also have their own place to chat. Our Club TWiT also have their own place to chat, our Club TWiT Discord, which is a place going 24/7. In fact, we have a great coding section in there. We're talking about this kind of stuff all the time.

If you want to get copies of the show, there's two places you can go. Of course, our site, TWiT.tv/sn. Steve's got them, too. In fact, he has two unique formats. He's got the 64Kb audio, just like we do. But he also has a 16Kb version for people who just don't want a big file, for the bandwidth-impaired. He also has transcripts written by the great Elaine Farris. She writes it all down so you can read along as you're listening, or even just read instead of listen. Or, and the most useful thing, search the transcripts to find the part of the show that you're interested in. And that goes - you don't have transcripts for all the shows. You just have it for...

Steve: Yeah, we went back and did it from day one.

Leo: Wow. Wow. That's really nice. Thank you. That's all at GRC.com. While you're there, by the way, you might want to check out SpinRite. That's Steve's day job, his bread and butter, the world's best mass storage recovery and maintenance utility.

You can pick up a copy of 6.0 right now. You'll get a free copy of 6.1 when it comes out if you buy today. You'll also get to participate in the development of 6.1. Steve's working very hard on that. It's imminent. GRC.com. Plus lots of free stuff like ShieldsUP! and so forth.

And InControl, which I plugged on the radio show this weekend because somebody said, "I don't want it. I never want to use Windows 11." I said, "Well, I've got a perfect program for you. Be in control." I guess that's all I need to say except thank you again, Steve, for doing such a great job, and we'll see you next time on Security Now!.

Steve: Always. Right-o. What is it, one more, we've got one more episode this month because we got normal-size month, and we started on the first of the month. So on the 29th.

Leo: Spring is here. The spring edition.

Steve: Right-o. Bye.