Memory-Safe Languages

Description: This week we have another event-filled Patch Tuesday retrospective. We look at a newly published horrifying automated host attack framework which script kiddies are sure to jump on. We have a welcome new feature for GitHub, crucial vulnerabilities in the LiteSpeed web server, a spiritual successor to TrueCrypt and VeraCrypt for Linux, Australia's announcement of their intention to proactively attack the attackers, a controversial new feature in iOS 16.1.1, a couple more decentralized finance catastrophes, some miscellany and listener feedback. Then we'll finish by looking at a just-published advisory from U.S.'s National Security Agency, our NSA, promoting the use of memory-safe languages.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-897.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-897-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. It's another event-filled Patch Tuesday retrospective. We'll talk about a new automated attack framework the script kiddies are going to just love, a replacement for TrueCrypt for Linux that gives you plausible deniability, and then he'll take a look at the NSA's proposal for memory-safe languages. This is one everyone should pay attention to, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 897, recorded Tuesday, November 15th, 2022: Memory-Safe Languages.

It's time for Security Now!, the show where we talk about your security and privacy online with our guy right here, this cat, Mr. Steven Gibson of the GRC.

Steve Gibson: We will protect you whether you want it or not.

Leo: Hi, Steve. Happy Tuesday. We survived Election Day, and we are now on to other things.

Steve: Well, we've got tonight's announcement from Mar-a-Lago.

Leo: Oh, that's right. It's 'uge. A 'uge announcement.

Steve: It will be at 6:00 p.m. Pacific time, 9:00 p.m. Eastern for those who are interested. I'm certainly a spectator.

Leo: I will be watching, yeah.

Steve: I will be glued with my popcorn, and we'll see what happens. It's a lot of fun.

Leo: Yes.

Steve: Also fun is today's topic. This is another one of these weeks where there was no, like, major cyber research that was - I always like to, you know, to track cyber research things. Nothing really happened except a bunch of news. Except that the NSA published a real interesting sort of an appeal to everyone to seriously consider switching to the use of memory-safe languages.

Leo: Yes. Yes.

Steve: Yes. Thus the title of today's episode of Security Now! 897 for November 15th, "Memory-Safe Languages." We're going to do a little retrospective on the event-filled, more eventful than Microsoft would have wished, Patch Tuesday. We're going to look at a newly published horrifying automated host attack framework, which script kiddies are sure to be jumping on. We've got a welcome new feature that's been introduced into GitHub. Also three critical vulnerabilities in the - it's only sixth in the world, but that gives it 1.9 million instances, the LiteSpeed web server.

Leo: Oh.

Steve: We've also got something I think is going to be of interest to our Linux users, the spiritual successor to TrueCrypt and VeraCrypt for Linux. And then we're going to do a little segment, 2.5 minutes we're going to play of Australia's announcement of their intention to proactively attack the attackers.

Leo: Oh, that doesn't sound good.

Steve: I don't know how you could do that legally.

Leo: Yikes.

Steve: It's like, oh, okay. And boy, she's pissed off. We've also got a controversial new feature that was added to iOS 16.1.1. Also I just have to touch on a couple more decentralized finance catastrophes, believe it or not, Leo, other than the collapse of FTX last week, or weekend before last. We've also got some miscellany and listener feedback, and then we're going to wrap up by taking a look at this National Security Agency promotion of the use of memory-safe languages. So I think another great podcast for our listeners. Oh, and because of last Tuesday's election, they put up a chart on the screen

at one point where I just said, okay, stop everything, hold on. This is just so wrong. And we probably talked about this last year. But again, I just can't resist because this just - it just makes my blood boil. It's like, who's in charge?

Leo: You know, I'm looking at it. I'm trying to figure out what you don't like about it. We'll find out in just a little bit. Oh, I see. Yes, I do see.

Steve: It's green. Green is like way bigger than orange.

Leo: Yes. You're right. I see. We'll show you in a second. Okay. Now that I look at it, yes. Interesting, huh. It's a perspective thing. You just wouldn't understand, Steve. It's all about TV production.

Steve: I lack a perspective, that's right.

Leo: All right. I've got a chart.

Steve: So the whole point of a pie chart...

Leo: Yes.

Steve: ...is to show you visually...

Leo: Yes.

Steve: ...the angles of the various slices of the pie. I mean, the reason aircraft instrumentation and even our cars still use dials, still have pointers, and a clock face, an analog clock face is so powerful, is that it requires no interpretation, no figuring out what the numbers mean in order to get meaning from this. They tried to go all digital in cockpits years ago, and it turns out that was not a good idea because a pilot can instantly see the angle of needles as opposed to needing to interpret numbers.

Leo: Yeah. Yeah.

Steve: So what happened last Tuesday during the election is that the various outlets could not resist doing this 3D thing to their pie charts. And in order to make it more fancy 3D, they accentuated the third dimension, creating this perspective. Well, so today's Picture of the Week shows a random exit poll with five different values. And the largest one is smaller looking than the next largest one. In this chart green looks like it's way more than 90 degrees, when in fact it's not. I mean, compared to the orange thing. Anyway, it just - I know I've mentioned this before. But I saw this, and I thought, oh, you know, come on, guys. Just give us angles which are a flat pie chart that we're looking down on from above, rather than like, oh, we're going to make this 3D and fancy.

Leo: But it's pretty, Steve.

Steve: Yeah.

Leo: It may be inaccurate, but it's pretty.

Steve: Yeah, that's right.

Leo: Yeah.

Steve: Okay. Anyway, I'm done with my rant on that. Last Tuesday was November's, not only was it the midterm elections in the U.S., it was November's Patch Tuesday. We had security updates being released by Adobe, Microsoft, SAP, Android, VMware, Citrix, and others. Everybody sort of piled on, probably hoping to like not be seen as obviously, if everyone does it on the same day. In the case of Microsoft, 67 security flaws were fixed. So, you know, on the high side, but not 167. And among those were the two lingering so-called "ProxyNotShell" zero-day vulnerabilities which have been continuing to plague those running onsite Exchange Server installations. There were also another four actively exploited zero-days fixed by last week's update. There was a remote code execution in the JScript9 scripting language. The Google TAG team found and reported that to Microsoft. And again, zero-days; right? They were being exploited when they were discovered.

There was a Mark-of-the-Web bypass known as "ZippyReads." An Elevation of Privilege in the Windows Print Spooler service, we're still discovering problems with that. Microsoft found that one being exploited. And also an elevation of privilege in the Windows cryptographic number generation key isolation service which, again, clever people are finding these problems and using them to elevate their privileges, typically obviously bad people. So Microsoft found that being happening.

However, amid all this, we did not get past this month's Patch Tuesday without the induction of some new headaches for enterprise admins. Since last Tuesday, Microsoft has been investigating the cause behind many reports, and no doubt many quick rollbacks of the month's "improvements" to Windows, which affected enterprise domain controllers, which started experiencing Kerberos sign-in failures and other authentication problems.

Well, we now know since Microsoft said, well, we're investigating this, we know what's going on. A little bit about Kerberos. After Windows 2000, Kerberos replaced the creaky and never quite ready for prime time homegrown NT LAN Manager protocol. And we've discussed NTLM's many security problems through the years on this podcast. Kerberos was designed by MIT and was first released, interestingly, the year after I first published SpinRite. Yes, way back in 1988 was when Kerberos happened. So it's been around for 39 years. It's a bulletproof client server mutual authentication protocol that works. Well, at least it did for Windows until last week. They're going to get it fixed, I'm sure.

They've acknowledged the trouble by saying - this is Microsoft: "After installing updates released on November 8th, 2022 or later on Windows servers with the domain controller role, you might have issues with Kerberos authentication." Which is to say, you know, pretty much all authentication that the domain controller's using. They said: "When this issue is encountered, you might receive a Microsoft-Windows-Kerberos-Key-Distribution-

Center Event ID 14 error event in the System section of event log on your domain controller with the below text.

"And as a consequence, here's what happens. Domain user sign-in might fail. This also might affect Active Directory Federation Services authentication. Group Managed Services Accounts used for services such as Internet Information Services" - their IIS web server - "might fail to authenticate. Remote Desktop connections using domain users might fail to connect."

Of course, all of these "mights" are actually "wills." Right? "You might be unable," and we read this as you will be unable, "to access shared folders on workstations and file shares on servers. Printing that requires domain user authentication might fail," or will. Anyway, so, you know, this is the typical collapse that we are with increasing frequency seeing following the application of a monthly round of patches and improvements. Thanks to a listener, Jonathan, I'm sorry, Johan Mellberg, who tweeted about this, there's a Reddit thread that explains what happened. Now, the current best practice is to disable the use of the very old RC4 cipher for Kerberos. But November's trouble has hit those enterprises which did. So this suggests that Microsoft did not test their November changes under what is now well-established current best practices.

I've got a link in the show notes to this Reddit thread for anyone who's interested and who would be affected by this. It documents the fixes to this which have worked for all of those who have implemented them. So that might be useful to some of our listeners. We tell everyone, of course, that patching as soon as possible can often be critical. We know that it can be. But it's also true as we seem to keep seeing that doing so might also cause major systemic failures throughout an organization. So it's no wonder that those whose job this is to deal with this are notoriously high stressed.

Anyway, so a bunch of problems fixed, a bunch of zero-days fixed, a regression which Microsoft will fix, I'm sure, and some workarounds. So your typical Patch Tuesday.

This next bit of news falls under the category labeled "Not all things that can be done should be done." And we occasionally encounter those on the podcast. Unfortunately, a new, smart, host attack automation framework - yes, a host attack automation framework - was open sourced a week ago, last Tuesday, on GitHub. My first thought was, "Please don't make attacking hosts any easier for the script kiddies." But it's too late.

A cybersecurity engineer by the name of Mazin Ahmed explained that back in 2019 he and a fellow researcher, Khalid Farah, participated in a security competition for developing offensive security tools. Not defensive, offensive. And as we'll see a bit later, these might be now of interest to Australia, but we'll get there in a minute.

So Mazin wrote, he said: "I enjoy building security tools, and this competition was funded by HITB" - that's Hack In The Box - "with a reward of $100,000 for the winners." He says: "I thought it would be an interesting challenge to work on as a side project. I met my friend Khalid. He was also interested in winning this competition. We signed up and, once accepted, we started meeting regularly to build this project."

Okay. So the Hack In The Box CyberWeek AI Challenge, as it was called, had two categories, Host Exploitation or Malware Evasion. And unfortunately they chose the Host Exploitation option. So he wrote: "Host exploitation sounds more relevant to our experience. The goal was to build a host exploitation framework using AI, based on the concept of DeepExploit. The winning team should ideally prove the accuracy of the model, the improvement of the training and execution speed, and the technical features that have been added to the framework. We started experimenting with DeepExploit and

how it works, and we decided to start a new project based on the ideas we had on how to improve that tool.

"This eventually ended up having us develop [something that they named] Shennina (S-H-E-N-N-I-N-A), a host exploitation framework." He said: "The project is four times faster than DeepExploit. We were excited about the results.

"Shennina comes with a deception detection capability that detects if the machine being exploited is a Virtual Machine or a Container, and then terminates post-exploitation once it's detected. This feature is powered by Metasploit modules. The Shennina Framework has qualified for the top five projects out of 40," that is, in that competition. He said: "We worked on developing the tool further to prepare for our final demo that will be live at HITB Abu Dhabi 2019."

Okay. So that was, what, three years ago. He said: "Unfortunately, the rules of the competition and the judging criteria changed during the demo day." Which I guess meant that they didn't do as well as they expected to. He said: "We enjoyed HITB CyberWeek 2019. It was an amazing journey, and I enjoyed building the Shennina Framework. I also presented my research on JWT hacking at the time." JWT is the abbreviation for JSON Web Tokens. And he says: "It was a busy week. Finally," he said, "We are planning to open-source the project and the experiment. There are no plans for further maintaining Shennina in the near future."

And that open sourcing of Shennina just came to pass. This powerful Shennina Host Exploitation framework is now up on GitHub. They do comment under the FAQ question, they ask themselves: "Why are we solving this problem with AI?" And they reply: "The problem should be solved by a hash tree without using AI. However, the HITB CyberWeek AI Challenge required the project to find ways to solve it through AI." So that's what they did.

Okay. So extracting some of the more interesting bits from the project's abstract, they explain: "Shennina is an automated host exploitation framework. The mission of the project is to fully automate the scanning, vulnerability scanning and analysis, and exploitation using artificial intelligence. Shennina is integrated with Metasploit and Nmap for performing the attacks, as well as being integrated with an in-house command-and-control server for exfiltrating data from compromised machines automatically."

And I'm thinking, "And why post this tool publicly?" Wow. Then they continue: "Shennina scans a set of input targets for available network services, uses its AI engine to identify recommended exploits for the attacks, then attempts to test and attack the targets. If the attack succeeds, Shennina proceeds with the post-exploitation phase. The AI engine is initially trained against live targets to learn reliable exploits against remote services. Shennina also supports a heuristics mode for identifying recommended exploits. The documentation can be found in the Docs directory within the project."

And then they enumerate their features: automated self-learning approach for finding exploits; high performance using managed concurrency design; intelligent exploits clustering; post exploitation capabilities; deception detection; ransomware simulation capabilities; automated data exfiltration; vulnerability scanning mode; heuristic mode support for recommended exploits; Windows, Linux, and macOS support for agents; scriptable attack method with the post-exploitation phase; exploits suggestions for Kernel exploits; out-of-band technique testing for exploitation checks; automated exfiltration of important data on compromised servers; reporting capabilities; coverage for 40-plus TTPs - that's of course Tactics, Techniques, and Procedures - within the MITRE ATT&CK Framework; and supports multi-input targets.

Well, that's just great. It's bad enough - really. I mean, do we have to be doing this, making this so easy for the script kiddies?

Leo: This is the excuse they always give. It's for pen testing. You know, it's to simplify your pen testing.

Steve: That's right. It's bad enough that the quite sophisticated and powerful Nmap scanner and Metasploit exploit frameworks, both which have been developing and evolving over many years, already allow sophisticated hackers to scan network regions and exploit them. But now this process has been automated with a somewhat smart frontend that incorporates Nmap and Metasploit to create a turnkey, all-in-one automated system that can be launched by junior hackers who would never have the ability to create such a tool themselves. So buckle up.

Leo: You think it's meant for malicious use or - obviously you do.

Steve: I just think, like, you know, it's one of those things that just because you can doesn't mean you should. This will be of vastly more interest to script kiddies who say, oh, look, we download it, and I run it on my Linux box, and it just attacks everything and exploits them.

Leo: Oh, look. It's so easy.

Steve: Yeah.

Leo: Yeah. But, I mean, you can also see how red teams need tools or would like tools like this.

Steve: Yes. So I'm sure that a red team could point it at their own network and discover useful things. Unfortunately, it also accepts the command-line star dot star.

Leo: Oh, ow.

Steve: Which, you know, put it on the Internet and let it loose.

Leo: Yeah. Yikes.

Steve: So, and the fact that it runs on Linux means that if you compromise routers, routers are all Linux-based now. So just host this and start scanning and exploiting. Wow. Yeah.

So again, we've seen where there's a problem with people posting proofs of concept to GitHub now because they're immediately weaponized. Well, now we've just weaponized Nmap and Metasploit so you don't even have to read the manuals for them. You just press Go. And unfortunately, people will. Wow.

Leo: Ah, well.

Steve: You know, one thing we've seen, Leo, over and over, is the only way to drive change is to force it to happen. So the upside is, after a period of cataclysm, we may end up with stronger software as a consequence, just because we didn't have a choice because, you know, these AI bots are now roaming around intelligently learning how to attack. Skynet, anyone? Wow.

Okay. In happier news, GitHub has just added a new private vulnerability reporting feature. It allows security researchers to report vulnerabilities which they have found which exist in public repositories to their respective owners via a private channel for the first time. So in other words, this new feature will allow sensitive security-related reports to be filed to repo owners without having to file a publicly viewable entry in a repo's issues tracker.

Leo: Probably should have had that a long...

Steve: That was my first thought was, oh, you're celebrating that in 2022? Why is this a new feature today? Yeah, this should have been done long ago. But I guess maybe it wasn't in keeping with the spirit. Again, this is - we need some evolution of the spirit; right? And we're going to get that here before long.

As I mentioned, there are presently 1.9 million unique instances of the LiteSpeed web server providing services on the Internet. LiteSpeed is ranked sixth most popular web server, which gives it about a 2% share of all web servers globally. So it's not, you know, not a big name, but it's there. The problem is, after some research by Palo Alto Networks' Unit 42, any editions of either the free and open source version, which is known as OpenLiteSpeed, or the enterprise version, which is just LiteSpeed, which have not been updated in the past month, need to be patched immediately. So I'm bringing this to our listeners' attention because among them, I mean, I remember looking at LiteSpeed when I was considering servers. It's a very - it's a recently written, fast, fleet little server. But it's got some problems.

The Unit 42 research team took a close look at the source code for OpenLiteSpeed, which of course as you are often telling us, Leo, and you're right, a big advantage of open source, right, is there it is.

Leo: Yeah.

Steve: Researchers, if they're so inclined, can look at it and go, uh, wait a minute, that doesn't look like a type safe cast there and so forth. Anyway, the Unit 42 guys from Palo Alto Networks took a look at it and discovered three different vulnerabilities in the web server. The three vulnerabilities have been confirmed to similarly affect the non-open source enterprise version, which obviously is based on the same code base. By chaining and exploiting the vulnerabilities, adversaries can compromise the web server to gain fully privileged remote code execution. Meaning remote, like over across the Internet. So the three vulnerabilities, there's a Remote Code Execution which is rated high severity, with a CVSS of 8.8; and a Privilege Escalation, same, high severity, 8.8; and a Directory Traversal which is given a medium severity of 5.8.

A little over a month ago, Unit 42 responsibly disclosed their discovery of these three vulnerabilities to the server's publisher and their maintainer, known as LiteSpeed Technologies, and even provided suggested remediation. Okay. That was on October 4th. Two weeks later, on the 18th, LiteSpeed Technologies released their updates. So anyone using the open version from 1.5.11 through 1.7.16 needs to immediately update to 1.7.16.1. And LiteSpeed versions 5.4.6 through 6.0.11 need to immediately move to 6.0.12.

Again, those were released on October 18th, not quite a month ago. So if this affects you, if you've got LiteSpeed running, and you haven't, and you don't know that you've updated in the last month, do it. There are 1.9 million of these in use. And since the fixes will be reflected in the open source version, bad guys will have no trouble seeing what was fixed and designing an exploit chain that Unit 42 warned of. Since the location of all LiteSpeed web servers are known to Shodan and other Internet scanners, these servers will be found. So don't delay.

Okay. A really cool bit of news and of interest for Linux users. Last Thursday, the guys at Kudelski Security Research open sourced an interesting utility that I think will be of interest to our listeners. Here's how Kudelski explains their creation. They said: "Today we are excited to release Shufflecake, a tool aimed at helping people whose freedom of expression is threatened by repressive authorities or dangerous criminal organizations, in particular whistleblowers, investigative journalists, and activists for human rights in oppressive regimes. Shufflecake is FLOSS (Free/Libre Open Source Software). Source code in C is available and released under the GNU General Public License 3 or superior."

They explain: "Shufflecake is a tool for Linux that allows the creation of multiple hidden volumes on a storage device in such a way that it is very difficult, even under forensic inspection, to prove the existence of such volumes. Each volume is encrypted with a different secret key, scrambled across the empty space of an underlying existing storage medium, and indistinguishable from random noise when not decrypted.

"Even if the presence of the Shufflecake software itself cannot be hidden, and hence the presence of secret volumes is suspected, the number of volumes is also hidden. This allows a user to create a hierarchy of plausible deniability, where 'most hidden' secret volumes are buried under 'less hidden' decoy volumes, whose passwords can be surrendered under pressure. In other words, a user can plausibly lie to a coercive adversary about the existence of hidden data by providing a password that unlocks decoy data. Every volume can be managed independently as a virtual block device, in other words, partitioned, formatted with any file system of choice, and mounted and dismounted like a normal disk. The whole system is very fast, with only a minor slowdown in I/O throughput compared to a bare LUKS-encrypted disk, and with negligible waste of memory and disk space."

And I love this. They said: "You can consider Shufflecake a spiritual successor of tools such as TrueCrypt and VeraCrypt, but vastly improved. First of all, it works natively on Linux, it supports any file system of choice, and can manage up to 15 nested volumes per device, making deniability of the existence of these partitions highly plausible." So for anyone who's interested, look at Shufflecake.net, S-H-U-F-F-L-E-C-A-K-E dot net. And you'll find all the information there.

Leo: Yeah, it looks really good. For plausible deniability, I love it.

Steve: Yeah, they did a beautiful job.

Leo: Yeah. I have your video ready, Steve.

Steve: So, good. Earlier I mentioned that Australia might have an interest in offensive as opposed to purely defensive cyber weaponry. As we've previously noted here, Australia has been recently having more than their share of problems with cyberattacks. They've had a slew of them, and they're apparently getting a bit tired of them. So the straw that may have broken the camel's back is the most recent ransomware cyberattack on an Australian private insurance provider named Medibank. Medibank said that attackers broke into their network, stole internal files - including sensitive personal and healthcare details on 9.7 million Australians - then encrypted their files and demanded a ransom.

When Medibank refused to pay, the ransomware gang, going by the name of BlogXX, although believed to be a spinoff of REvil, began leaking some of Medibank's patient records to intimidate Medibank and create public pressure to pay the ransom. One of the group's first leaks was a file named "abortions." Not surprisingly, the BlogXX group's actions have galvanized the Australian government, and Australia has now vowed to go on the offensive. The group's activities have been met with outrage and significant political attention from the highest levels of the Australian government.

Okay. So Clare O'Neil is an Australian Minister for Home Affairs, and she's the Minister of Cyber Security. She was interviewed last week by ABC Insider and posted her 2.5-minute discussion to Twitter. So here's what Clare had to say.

[BEGIN CLIP]

CLARE O'NEIL: Mark Dreyfus and I announced with the Deputy Prime Minister yesterday that we're setting up a permanent standing operation, a partnership of new policing between the Australian Signals Directorate, which are the [indiscernible] of the Australian Public Service and the Australian Federal Police. This is an entirely new model of operating for these two organizations. What they will do is scour the world, hunt down the criminal syndicates and gangs who are targeting Australia in cyberattacks, and disrupt their efforts. This is Australia standing up and punching back. We are not going to sit back while our citizens are treated like this way and allow there to be no consequences for that.

INTERVIEWER: Okay. Just to be clear, though, this task force has been in place for while, though, hasn't it?

CLARE O'NEIL: No, that's incorrect. This is a new operation, a permanent standing force of 100 of the best, most capable cyber experts in this country that will be undertaking this task for the first time, offensively attacking these people, David. So this is not a model of policing where we wait for a crime to be committed and then try to understand who it is and do something to the people who are responsible. We are offensively going to find these people, hunt them down, and debilitate them before they can attack our country.

INTERVIEWER: What's your expectations here of what they'll be able to achieve? Because, you know, we know whether it's in this new standing operation or in the previous taskforces, they've been trying for a while to go after the attackers. The Americans have for years. They did manage to arrest a few of them about a year ago. What's your expectation of what they'll realistically be able to achieve?

CLARE O'NEIL: Yeah. Well, I think there's a perception in the community that it's hard to do anything about cyberattacks, and that's actually wrong. There's an enormous amount that we can do. I think we need to shift away from the sense that the only good outcome here is someone behind bars because that can be hard when we've got people who are

essentially being harbored by foreign governments and allowed to continue this type of activity. But what we can do is two really important things. The first thing is hunt these people down and disrupt their operations. It weakens these groups if governments like ours collaborate with the FBI and other police forces and intelligence agencies around the world. But the second important thing that we need to do is stand up and say that Australia is not going to be a soft target for this sort of thing. And if people come after our citizens, we are going to go after them.

[END CLIP]

Leo: I don't think that's legal; right? I mean, you can't pursue people before they commit the crime, for one thing.

Steve: I know. I mean, this is like, what?

Leo: Very Australian. Very Australian, yeah.

Steve: Yeah, well, so her statement was released the same day that the Australian Federal Police issued a statement formally identifying the Medibank hackers as being located in Russia, which, you know, comes as a surprise to no one. Australian Federal Police Commissioner Reece Kershaw said in his statement: "We believe we know which individuals are responsible, but will not be naming them. What I will say is that we'll be holding talks with Russian law enforcement about these individuals."

And, you know, despite an explosion in ransomware attacks, as we noted last week, more than $1.2 billion in payments made in 2021 alone, the cases where law enforcement agencies fought back and hacked the hackers have been rare. There have been a few success stories, such as the DOJ, the U.S. Department of Justice, managing to recover the Colonial Pipeline payment that was made to the DarkSide group. And both U.S. CYBERCOM and the FBI hacked REvil's servers following the Kaseya-based attacks. In both cases, DarkSide and REvil's operations were shut down following these proactive, potentially, I guess, offensive operations. And these effective responses did appear to give attackers at least a bit of pause. Yet the problem obviously persisted, and persists. So it's going to be very interesting to see what might come from Australia's new "we're not going to take this lying down anymore" stance.

But exactly as you said, Leo, until now the bad guys have had the advantage that they don't care about breaking cyber intrusion laws, whereas global governments have been hampered by their need to play by the rules and act within the law. So, you know, what does Australia exactly mean when they say they're going to be going after the bad guys and disrupt them? Are they going to break cyber intrusion laws to get that done? I guess we'll see. Wow.

Leo: She may have misunderstood exactly what they were going to be doing. Well...

Steve: I don't know because, I mean, he sort of pushed back, and she pushed back on him.

Leo: Yeah. Yeah. Wow.

Steve: So they did celebrate Medibank's decision to not capitulate to the demands of its attackers. And separately, O'Neil suggested that the government might look into a law which formally bans payments made to ransomware and data extortion attackers altogether, with the hope that this would strangle the financial incentive behind those attacks. Like you're not going to get any money out of anyone you attack in Australia, so go attack other people. Again...

Leo: I mean, I understand the outrage.

Steve: Oh, yes.

Leo: You know, it's infuriating. But...

Steve: I know.

Leo: But.

Steve: You can't be a vigilante.

Leo: Yeah, you can't, no.

Steve: So Apple iOS. What observers are saying looks like Apple capitulating to demands from China, the latest iOS 16.1.1 and also in the next beta of 16.2, have added a new timeout to the AirDrop sharing feature which limits the "Share with Everyone" option to 10 minutes. The reason Chinese influence is suspected is that, for the time being at least, this only affects iPhones purchased in China. This new restriction is tied to the hardware.

On the other hand, it's also worth noting that many people outside of China have celebrated the addition of this restriction as an extremely useful security improvement, since inadvertently leaving "Everyone Sharing" on, you know, enabled, could represent a significant security risk.

And interestingly, this is not the first time Apple has customized some aspects of their iPhone offerings for China. In researching this a bit, I learned, believe it or not, that the Taiwanese flag emoji is not available on iPhones sold in China. Talk about petty. Anyway, Apple also uses the hardware tied method to limit the volume level of its devices within the EU, as is required by the laws there.

So in the case of this 10-minute time limit, the motivation appears to be due to the fact that Chinese protesters had been using the feature to spread posters in opposition to Xi Jinping and the Chinese government. And for those who think that this is a useful feature, Bloomberg reported that Apple does have plans to make this new AirDrop option available globally next year, though presumably optional as opposed to mandatory, where it is now in China.

Again, you know, we've seen that these global companies, whether it's Apple or Google or whomever, you know, if they want to operate within other countries that have different regulations, they need to alter their behavior, if they want to stay there.

Okay. A couple of Decentralized Finance notes that I just, I couldn't help with because these are just, again, it must be that by now our listeners understand where not to put their money. The DeFi platform Pando said it was the target of a hack Saturday, actually Saturday before last, when a threat actor attempted to steal more than $70 million worth of cryptocurrency from the platform's wallets. Pando said that it managed to hold onto $50 million of the attempted stolen funds, but the attacker successfully stole more than $21.8 million of its funds. They said that the hacker used an Oracle attack against one of its protocols, and that they're still hoping to negotiate with the attacker to return some of the stolen funds. To which apparently the proper response is, well, you can hope.

I wondered whether that had happened, since now that's been 10 days. So far it doesn't appear the case. I found Pando's Twitter thread where they had announced the hack of their system. Then over the course of several days they followed the stolen funds as they were anonymously moved from one currency or exchange to another. You know, this is just a mess. Imagine watching nearly 22 million of your dollars moving around and being able to do nothing about it. Such is the world we're in today.

Also, the DeFi platform DFX Finance suffered a crypto heist, reporting the loss of $4.185 million worth of cryptocurrency assets following an attack on its platform last week. The company said the incident was identified as a - here's a new one - a reentrancy attack, which sounds like another of those increasingly common, "Oh, shoot, we failed to consider that possibility when we were designing our platform. There went more than $4 million."

So anyway, as I said, I'm sure everyone who follows the podcast by now knows that no one should rely upon what over and over again appear to be half-baked, poorly conceived, decentralized finance systems. And it clearly doesn't matter how well intended the creators and the operators of these systems are. Remember the famous Willie Sutton quote, you know, where he explained that the reason he robbed banks was because that's where the money was. For many people, money is the great motivator. That's the singular thing that's driving the ransomware attacks today. They could care less about anyone's data. That's of no interest to them. They want money. And these DeFi systems are apparently sloshing around in way more money than they are responsible enough to manage. So, yeah, don't put yours there.

A little bit of miscellany. It turns out that a product which was a past sponsor of TWiT and of the Security Now! podcast, The Helm, was unable to survive changes resulting from COVID-19. They recently announced that they would be unable to continue operating past the end of the year. So they provided their users with 60 days' notice, so November and December, and have already stopped accepting new subscriptions. They explained that in 2019 they had relocated their production operations from Mexico to China in order to improve their scalability, and began to work on a v2 system. But then COVID hit, and their scheduling times more than doubled.

I'm bringing this up here for two reasons: First, anyone who had obtained their own domain that was hosted at Helm's registrar, you may wish to move it to somewhere else if you want to hold onto it. You know, like I would suggest Hover, my own registrar before they also became a TWiT sponsor. So current users will have only until the end of this year to get their domain moved. But The Helm is fully supporting that movement and will work with any of their current users to do so.

The second reason I'm mentioning this is that on their FAQ page, in response to the question "What can I do with the Helm server after you shut down?" they wrote: "We are working on a firmware update for Helm servers to be converted to Linux servers running Armbian. We will provide documentation for this conversion, along with pointers to some guides for running mail and file-sharing services if you would like to continue using your Helm. We expect this firmware update to be available in early to mid-December." So I've

got a link here in the show notes to the support page about this happening, about Helm shutting down. So, you know, just a heads-up to anyone listening that, if you are a Helm user, some action will need to be taken between now and the end of the year. I would imagine you'll want to keep your eye out for this firmware update to update to your device that would make it freely useful elsewhere.

Leo: Yeah, I think they're doing the best they can.

Steve: Yeah, yeah.

Leo: They were hit by supply chain shortages and so forth. I know the guys. They're great people. And they're just trying to do the best thing they can. It won't be an email server anymore because serving your own email is not something for the faint of heart. It's very difficult to do from a home network. Even for us it would be difficult to do because of the fact that Gmail, the big mail companies just don't want to hear from you. They'll just blackhole you like that.

Steve: Yeah.

Leo: But at least you'll be able to use it for something. And it's a cool - I still have mine. It's a cool little thing. I'll put this, what is it, ArmDebian?

Steve: Armbian.

Leo: Armbian. Not Ambien. Armbian.

Steve: Right. Okay. So Elon meets Twitter. There's no way for a weekly podcast to possibly follow the insanity that has befallen Twitter ever since Elon has taken the reins. But one of our listeners, Ed Ross, put me onto a Twitter thread from yesterday that gave me a chuckle, which I thought our listeners would appreciate. So Elon says, he announced yesterday: "Part of today will be turning off the microservices bloatware." He says: "Less than 20% are actually needed for Twitter to work!"

Leo: And then of course the engineers who said, "Elon, you don't understand how it works," were fired immediately.

Steve: Uh-huh. Yes, exactly. And after the results of pulling those plugs resulted in a new catastrophe du jour, someone named Zach Silberberg tweeted. He said: "Apparently they didn't turn off two-factor authentication, but they did turn off the service that sends you the two-factor authentication code. So if you log out and try to log in with an authentication code, you simply won't receive one."

Leo: Well, first of all...

Steve: Thank you, Elon.

Leo: You should have turned off SMS authentication a long time ago.

Steve: Yes.

Leo: Unfortunately, the way two-factor, you know, I went through this a couple of years ago on Twitter. The way two-factor works on Twitter, you have to turn it on first, but they give you two other methods, an authenticator app like Google Authenticator or hardware key. And once you get the others set up, you can uncheck, as I did, the SMS. So if you didn't do that, if you only had SMS, I don't know what you're going to do now. Don't log out. The other thing, Ed Bott did a good article on what to do now because Twitter security is clearly going to fail. He suggested, and I think he's right, to immediately go to your Twitter account and turn off all the third-party apps you've enabled. Most of us over the years have turned on a variety of tools, maybe even Login with Twitter and stuff. For instance, I get into Medium by logging with Twitter, so Medium shows up there. Given that we don't know what the security status of Twitter's going to be going forward, disconnect those. Right?

Steve: Very, very good point.

Leo: Just disconnect all of those. So I did that. Turn off two-factor with SMS, but make sure you have two-factor on because, you know. And then others are saying maybe you want to delete your DMs. Those are not encrypted. And if you had anything in there that you thought was private, maybe you should get rid of it now. It's inevitable at this point that it's going to slowly start to crumble. And the first thing that's going to go is security. So you'd be wise, I mean, if you want to stay on there, that's fine. But you'd be wise to give it nothing you don't want to lose.

Steve: What a mess.

Leo: Ugh, such a mess. It's sad. It's really sad.

Steve: Yeah, it is. Daniel Smith, a listener, he said: "This could be a new SN" - as in Security Now! - "saying." And then he quoted somebody on Twitter named Lewis Lovelock who replied to someone else. And I do love this quote: "There's nothing more permanent than a temporary solution."

Leo: It's true. You know that. You had a fix that you forgot you did; right? Ages ago to your server.

Steve: That's right. That's exactly right. You're right. I overrode DNS for my ecommerce. And then when they changed their IP, I stopped being able to accept payments. It's like, uh, whoops. Yeah.

Dennis Keefe, financial coach. He tweeted: "Hi, Steve. Wanted to let you know the sponsor link for Drata seems to send you to a 404. I'd hate for y'all to lose money. Love

the show." And so I went, I mean, I tried Drata.com/twit. And sure enough, it 404s. So I just thought, Leo, mostly for your benefit, that I would note that.

Leo: Yeah, they're a sponsor. Go to TWiT.tv/sponsors, and there should be a link on there to any sponsor. And if the one that we told you doesn't work - let me click Drata.com/twit. Error 404. So I will - whoops. Wait a minute. No, yeah, yeah. It says: "Things got out of control there. Automate compliance and risk management control past find this page control failed." Okay. I'll go, I'll be taking a walk. Well, no, after I do the next ad because I know you're getting close.

Steve: Yeah, yeah.

Leo: I will take a walk down to continuity and alert them. And thank you, Dennis, yeah.

Steve: Thank you for the pointer.

Leo: Yeah, I appreciate that, yeah. And by the way, if you want to tweet me, don't do it on Twitter. Do it on TWiT.social, our Mastodon instance.

Steve: There we go.

Leo: Yeah.

Steve: Neil Baldridge tweeted: "Hi, Steve. Just a quick thank you for passing along your recommendation for the Silver Ships series." He says: "I'm just into book 6, and finding the stories to be a great read. I'm trying to pace myself, but without much success. It's so nice to find an extensive series that's so enjoyable."

And so I'll just take this opportunity to note that I totally agree. The series author, Scott Jucha, suggests that after finishing book 13, the reader should switch over to the four-book "Pyreans" series. So that's what I did. I'm nearly finished with those four books. I'm more than halfway through the fourth of that four-book series. So I can attest that they are every bit as delicious as the Silver Ships series. Those four books trace the history of an entirely different Earth-descended colony, and its characters are every bit as interesting and alive as those in the Silver Ships.

And what's intriguing is that by this point the reader will have grown to intimately know two very different groups of human colonists, and Scott has explained that the two threads are going to be merging once the reader resumes with the Silver Ships series in book 14. I've come to know this diverse set of characters so well that I cannot wait to see what he has in store for them when they meet. It promises to be quite cool. So anyway, I'm glad that a lot of our listeners have picked up and are enjoying that series.

Tiemo Kieft tweeted: "Hey, Steve. I just wanted to respond to the whole OpenSSL debacle. You mentioned not many web servers accept client certs. This is not the case, though. Web servers serving big sites accept client certs because CDNs and DDOS protection services like Cloudflare use a client cert to assert their identity to web servers

they proxy. This feature is used to ensure that the server only responds to requests provided through Cloudflare. Cloudflare calls this feature 'Authenticated Origin Pulls.'"

So thank you, Tiemo. That certainly makes sense as a reason for web servers to solicit and accept client certs. I guess I still doubt that the number is huge, but I agree that it would definitely be another reason for some increased vulnerability to the recent OpenSSL v3 flaw. And of course it would be those servers that would be accepting client certs because they were being served by CDNs and DDOS protectors that are probably worth attacking. So that's a very good point.

And finally, Nariman Aga-Tagiyev, sorry about the way I pronounced your name, he said: "Dear Steve. After listening to Episode 289, whoa, sorry, Episode 892..."

Leo: You're just a little dyslexic, it's okay.

Steve: Dyslexic much? Yeah. Episode 892. He said: "I decided to replace my Raspberry Pi home VPN server with a ZimaBoard, which I ordered online right away. Following getting started instructions from the CasaOS, I created a new user and a complex password using its WebUI on port 80. But I was quite concerned when I noticed that the credentials set are only for the Casa web application. The board came with more than 10 default users on its Debian OS which still had their default passwords after the initial configuration steps." He says: "I could log in over SSH using casaos casaos for username and password. The root's default password is also casaos. There are no warnings or instructions on the ZimaBoard and CasaOS websites about how to remove default users and passwords from the board." He said: "I'm very concerned that if those boards are installed as a router according to this article" - and he provided a link - "anyone will be able to SSH to it if owners don't proactively delete all Debian users. Thank you for your podcasts. Your active listener for a few years already."

So I wanted to share that because that's definitely a big heads-up that would be important if somebody were using a ZimaBoard, not as I am, to run SpinRite and to develop SpinRite, but using it as a pfSense router and exposing SSH publicly where it's got the default casaos/casaos username and password. So thank you very much for sending that to my attention.

Leo: Yeah, that's nasty.

Steve: Yeah. And, boy, they really absolutely should make a note, or like the setup system should change username and password for all of those other users and not leave them - not only change them for the login to the WebUI and leave all of the other ones alone.

In a quick update about SpinRite, it's all but finished. I'm getting very, very close to the first pre-release alpha of 6.1. I ended up rewriting a bunch of the command-line parser to make drive selection extremely flexible and powerful. Then, as I had planned, I implemented the final verification for SpinRite by arranging to take a hash of an entire drive over which SpinRite had been run in order to absolutely verify that not a single bit had been changed across all of SpinRite's various machinations with the drive. And I discovered to my surprise that my AHCI driver would, in a very low but non-zero percentage of 16MB transfers, occasionally not properly read the drive's data.

To make sure this wasn't, like, I had just written this code. So to make sure that it wasn't the brand new hashing system that was at fault for this, I tried the same thing

with an IDE drive using SpinRite's also new Bus Mastering DMA driver, and that worked perfectly. So I know that my confidence testing code is working and that the new Bus Mastering driver is also working perfectly. But not so yet the new AHCI driver. This just happened on Sunday evening and also a little bit yesterday morning, you know, before I switched to working on the podcast. So I haven't had a second yet to dig into any of this.

My guess is that something is going on with my end-of-transfer status testing where I might be shutting down the controller before it's completely finished having transferred everything into RAM. Since it's transferring data asynchronously in the background using DMA, it's definitely necessary to make sure that it's finished. The good news is, thanks to now being able to take a hash of everything SpinRite reads, I'm able to test that quickly and verify any fixes that I make, and changes. So anyway, I'm anxious to get back to it there, as I will tonight, to see what's going on. And I expect to be able to say next week that I am finished with this round of SpinRite development, and that it's currently in alpha testing. So very, very, close.

Leo: Cool, cool.

Steve: And I'm very excited.

Leo: Yay.

Steve: And Leo?

Leo: Yes?

Steve: I'm going to wet my vocal cords, and then we're going to talk about memory-safe languages.

Leo: I'm very interested in this because, as you know, I like garbage collection. It's just me, but I like it. All right, Steve. Let's talk about memory.

Steve: Okay.

Leo: It is the number one problem, isn't it, in...

Steve: It is, absolutely. Like 70% of the vulnerabilities are caused by mismanagement of memory.

So what I'm going to share, verbatim, is what our National Security Agency, the NSA, published in an effort to improve security across the board. So they said - at the beginning there's a short executive summary. They said: "Modern society relies heavily on software-based automation, implicitly trusting developers to write software that operates in the expected way and cannot be compromised for malicious purposes. While developers often perform rigorous testing to prepare the logic in software for surprising conditions, exploitable software vulnerabilities are still frequently based on memory

issues. Examples include overflowing a memory buffer and leveraging issues with how software allocates and deallocates memory.

Microsoft revealed at a conference in 2019 that from 2006 through 2018, 70% of their vulnerabilities were due to memory safety issues. Google also found a similar percentage of memory safety vulnerabilities over several years in Chrome. Malicious cyber actors can exploit these vulnerabilities for remote code execution or other adverse effects, which can often compromise a device and be the first step in large-scale network intrusions.

"Commonly used languages, such as C and C++, provide a lot of freedom and flexibility in memory management while relying heavily on the programmer to perform the needed checks on memory references. Simple mistakes can lead to exploitable memory-based vulnerabilities. Software analysis tools can detect many instances of memory management issues, and operating environment options can also provide some protection; but inherent protections offered by memory-safe software languages can prevent or mitigate most memory management issues."

Okay. So that sets it up. Here's how they elaborate on this: "NSA recommends using a memory-safe language when possible. While the use of added protections to non-memory-safe languages and the use of memory-safe languages do not provide absolute protection against exploitable memory issues, they do provide considerable protection. Therefore, the overarching software community across the private sector, academia, and the U.S. government have begun initiatives to drive the culture of software development towards using memory-safe languages.

"How a software program manages memory is core to preventing many vulnerabilities and ensuring a program is robust. Exploiting poor or careless memory management can allow a malicious cyber actor to perform nefarious acts, such as crashing the program at will or changing the instructions of the existing program to do whatever the actor desires. Even unexploitable issues with memory management can result in incorrect program results, degradation of the program's performance over time, or seemingly random crashes.

"Memory safety is a broad category of issues related to how a program manages memory. One common issue is called a 'buffer overflow,' where data is accessed outside the bounds of an array. Other common issues relate to memory allocation. Languages can allocate new memory locations as a program is executing and then deallocate the memory, also called releasing or freeing the memory, later when the memory is no longer needed. But if this is not done carefully by the developer, new memory may be allocated again and again as the program executes. Consequently, memory is not always freed when it is no longer needed, resulting in a memory leak that could cause the program to eventually run out of available memory. Due to logic errors, programs can also attempt to use memory that has already been freed, or even free memory that has already been freed.

"Another issue can arise when languages allow the use of a variable that has not been initialized, resulting in the variable using the value that was previously set at that location in memory. Finally, another challenging issue is called a 'race condition.' The issue can occur when a program's results depend on the order of execution of two parts of the program accessing the same data. All of these memory issues are much too common occurrences.

"By exploiting these areas of memory issues, malicious actors, who are not bound by normal expectations of software use, may find that they can enter unusual inputs into the program, causing memory to be accessed, written, allocated, or deallocated in unexpected ways. In some cases, a malicious actor can exploit these memory management mistakes to access sensitive information, execute unauthorized code, or

cause other negative impacts. Since it may take a lot of experimenting with unusual inputs to find one that causes an unexpected response, actors may use a technique called 'fuzzing' to either randomly or intelligently craft multitudes of input values to the program until one is found that causes the program to crash.

"Advances in fuzzing tools and techniques have made finding problematic inputs easier for malicious actors in recent years. Once an actor discovers that they can crash the program with a particular input, they examine the code and work to determine what a specially crafted input could do. In the worst case, such an input could allow the actor to take control of the system on which the program is running.

"Using a memory-safe language can help prevent programmers from introducing certain types of memory-related issues. Memory is managed automatically as part of the computer language. It does not rely on the programmer adding code to implement memory protections. The language institutes automatic protections using a combination of compile time and runtime checks. These inherent language features protect the programmer from introducing memory management mistakes unintentionally. Examples of memory-safe languages include C#, Go, Java, Ruby, Rust, and Swift. Even with a memory-safe language, memory management is not entirely memory safe.

"Most memory-safe languages recognize that software sometimes needs to perform an unsafe memory management function to accomplish certain tasks. As a result, classes or functions are available that are recognized as non-memory safe and allow the programmer to perform a potentially unsafe memory management task. Some languages require anything memory unsafe to be explicitly annotated as such to make the programmer and any reviewers of the program aware that it is unsafe. Memory-safe languages can also use libraries written in non-memory-safe languages and thus can contain unsafe memory functionality. Although these ways of including memory-unsafe mechanisms subvert the inherent memory safety, they help to localize where memory problems could exist, allowing for extra scrutiny on those sections of code.

"Languages vary in their degree of memory safety instituted through inherent protections and mitigations. Some languages provide only relatively minimal memory safety, whereas other languages are very strict and provide considerable protections by controlling how memory is allocated, accessed, and managed. For languages with an extreme level of inherent protection, considerable work may be needed to simply get the program to compile due to checks and protections.

"Memory safety can be costly in performance and flexibility. Most memory-safe languages require some sort of garbage collection to reclaim memory that has been allocated, but is no longer needed by the program. There is also considerable performance overhead associated with checking the bounds on every array access that could potentially be outside of an array.

"Alternatively, a similar performance hit can exist in a non-memory-safe language due to the checks a programmer adds to the program to perform bounds checking and other memory management protections. Additional costs of using non-memory-safe languages include hard-to-diagnose memory corruption and occasional program crashes along with the potential for exploitation of memory access vulnerabilities. It is not trivial to shift a mature software development infrastructure from one computer language to another. Skilled programmers need to be trained in a new language, and there is an efficiency hit when using a new language. Programmers must endure a learning curve and work their way through any "newbie" mistakes. While another approach is to hire programmers skilled in a memory-safe language, they too will have their own learning curve for understanding the existing code base and the domain in which the software will function."

Now, here's something that we've never talked about before. But I think it's an important thing to mention here. They say: "Several mechanisms can be used to harden non-memory-safe languages to make them more memory safe. Analyzing the software using static and dynamic application security testing" - abbreviated SAST for Static Application Security Testing and DAST for Dynamic Application Security Testing - "can identify memory use issues in software. Static analysis examines the source code to find potential security issues.

Using SAST allows all of the code to be examined, but it can generate a considerable number of false positives through identifying potential issues incorrectly. However, SAST can be used throughout the development of the software, allowing issues to be identified and fixed early in the software development process. Rigorous tests have shown that even the best-performing SAST tools only identify a portion of memory issues in even the simplest software programs, and usually generate many false positives.

"In contrast to SAST, dynamic analysis examines the code while it is executing. DAST requires a running application. This means most issues will not be identified until late in the development cycle, making the identified problem more expensive to fix and regressively test. DAST can only identify issues with code that is on the execution path when the tool is run, so code coverage is also an issue. However, DAST has a much lower percentage of false positives than SAST. Issues such as a memory leak can be identified by DAST, but the underlying cause of the memory issue may be very difficult to identify in the software.

"Neither SAST nor DAST can make non-memory-safe code totally memory safe. Since all tools have their strengths and weaknesses, it's recommended that multiple SAST and DAST tools be run to increase the chances that memory or other issues are identified. Working through the issues identified by the tools can take considerable work, but will result in more robust and secure code. Vulnerability correlation tools can intake the results from multiple tools and integrate them into a single report to simplify and help prioritize analysis."

Couple last points. Anti-exploitation features. "The compilation and execution environment can be used to make it more difficult for cyber actors to exploit memory management issues. Most of these added features focus on limiting where code can be executed in memory and making memory layout unpredictable." Which of course we've talked about extensively before. They said: "As a result, this reduces a malicious actor's opportunities to use the exploitation tradecraft of executing data as code and overwriting a return address to direct program flow to a nefarious location.

"Leveraging options such as Control Flow Guard (CFG) will place restrictions on where code can be executed. Similarly, Address Space Layout Randomization (ASLR) or Data Execution Prevention (DEP) add unpredictability to where items are located in memory and prevent data from being executed as code. Bypassing ASLR and DEP is not insurmountable to a malicious actor, but it makes developing an exploit much more difficult and lowers the odds of an exploit succeeding. Anti-exploitation features can help mitigate vulnerabilities in both memory-safe and non-memory-safe languages."

So they conclude: "Memory issues in software comprise a large portion of the exploitable vulnerabilities in existence. NSA advises organizations to consider making a strategic shift from programming languages that provide little or no inherent memory protection, such as C and C++, to a memory-safe language when possible. Once again, some examples of memory-safe languages include C#, Go, Java, Ruby, Rust, and Swift. Memory-safe languages provide differing degrees of memory usage protections; so available code hardening defenses, such as compiler options, tool analysis, and operating system configurations should be used for their protections, as well. By using memory-

safe languages and available code-hardening defenses, many memory vulnerabilities can be prevented, mitigated, or made very difficult for cyber actors to exploit."

So, okay. I just wanted everyone to hear and to consider that. And for those of our listeners who may be in positions where a choice of implementation language can be made for applications where the program's security and integrity would be important, you know, consider perhaps breaking with the status quo that's been established and choose to bite the bullet and make a switch to a more secure programming language and environment. Take the relatively new Rust language, for example. Since its first release in January of 2014, Rust has been adopted by companies including Amazon, Discord, Dropbox, Facebook, Mozilla, Google, and Microsoft. And it's a systems implementation language. It does not slow things down.

Should changing to a memory-safe language not be feasible for whatever reason, at least consider looking into the idea of using those static and dynamic analysis tools against your existing code base. You don't have to treat what they say as gospel, but check out everything that they flag to make sure that it's not a problem. You may find that a chunk of those things are problems and be glad that you just ran this thing over your code to see what it thinks. All of that should really pay off.

Leo: Are all garbage-collected languages memory safe? In theory they would trigger an error at runtime if you addressed memory that didn't exist; right? They'd crash.

Steve: I think so, unless you used a pointer that was still available to you that no longer pointed to anything.

Leo: A compiler should catch that.

Steve: And we often see those use-after-free problems.

Leo: Yeah, right, right, right. Yeah. I mean, Rust actually doesn't do garbage collection, does something a little more complex. But Rust is very cool. Not a great language if you don't want to spend a lot of time learning it.

Steve: Yeah. I think Rust uses usage counting, as I recall.

Leo: Exactly. Yeah, reference/dereference, yeah.

Steve: Yes, yeah.

Leo: Count by reference. Sort of. It's kind of - it's really clever. It's unique, I think.

Steve: And we do hear that Rust has a steep learning curve.

Leo: Oh, it does. I tried. If you use Java, it should be pretty straightforward. If you're already using one of those languages that has a lot of declarative stuff, it should feel fairly comfortable to you.

Steve: Well, again, you know, C is showing its age.

Leo: Yeah.

Steve: C++ is a mixed blessing. Even its developer agrees that it sort of didn't work out the way...

Leo: If your language has malloc or calloc, that's a pretty good sign you should not be using it; right?

Steve: Yeah.

Leo: That's for sure. Very good. I think this is great. So did you write - it's almost like a column. Did you write this like a column? You should submit this to somebody.

Steve: Well, the NSA submitted it to the world.

Leo: Oh, it's the NSA. Ah, that explains it, yeah, yeah, yeah.

Steve: Yeah.

Leo: Because that's what it feels like. It feels like a column.

Steve: Yup.

Leo: I missed that one.

Steve: Yup, just wanted to share it because I thought it made a bunch of really good points.

Leo: Yeah, it's good, yeah. People forget the NSA has a couple of missions, one of which is to secure us at home through proper practices, whether it's hardening our machines or writing safe code. So that's a big part of the job, as well as spying on people and stuff like that.

Steve: Oh, that.

Leo: Sometimes, though, they're incompatible; right? You know, if you want to spy on them, you don't want them to use any secure. That's wrong.

Steve, great stuff, as always. Steve Gibson is at GRC.com. That's his website. Now, there's a lot there. So let me point you to a couple of things. I mean, it's just fun to browse around. But of course that's where you'll find SpinRite, which is Steve's bread and butter, the world's best mass storage maintenance and recovery utility, currently 6.0, soon to be 6.1. He's very close now. You'll get a free upgrade if you buy now.

Steve: Getting nervous, yeah.

Leo: Getting nervous?

Steve: Yeah, I'm getting nervous.

Leo: Do you have a button that you push that's like, Release the Kraken?

Steve: Well, there's just like so many little details; right?

Leo: I know.

Steve: But exciting. So I'm excited.

Leo: And, see, Steve - this is good, and you should understand this. Steve isn't one of those people who releases it, sees what the bugs are, and then fixes them. Steve wants to release a perfect piece of software from v1.0. So that would make me nervous, too.

Steve: In 2004 SpinRite 6 was released, and there are no known bugs.

Leo: Haven't patched it since. That's kind of amazing. Really amazing. So that's one thing, SpinRite. Probably GRC.com/SpinRite would probably take you there. You also should take a look at the podcast because he has two copies of the podcast that we don't have on our TWiT site. One is the 16Kb audio. It's a little scratchy. But if you have limited bandwidth, it's the smallest version. Actually, it's the second smallest version because the smallest version I'm sure is text only. That's the transcriptions Elaine Farris does, which are great. You might want them separately. You might want them together with the audio so you can read along as you listen. And it's certainly a utility you can use to search for part of any of the shows, any of the 897 shows, and find those comments right in the show. So it's really, really a great thing.

He also has 64Kb audio, the full quality audio. We have that and video at our website, TWiT.tv/sn. You can also find Security Now! on YouTube. There's a YouTube channel. Probably, I'm not sure, but I think it's YouTube.com/securitynow. If you go to YouTube.com/twit, you'll find, besides all the TWiT bits and the shorts and stuff, you'll find a list of all the shows, and you can subscribe from there, as well.

Of course the easiest thing to do probably for most people is to find a podcast player you like and subscribe to Security Now! on that. That way you'll get it automatically as soon as it's available. It downloads automatically. You can listen whenever you want.

If you want to watch us do it live, get the very first pressing, the EVOO Security Now!, you can - I never really thought of it that way, but I guess so - you can go watch us live at live.twit.tv every Tuesday, right after MacBreak Weekly. That time will vary, somewhere around 1:30 or 2:00 p.m. Pacific, 5:00 p.m. Eastern, 22:00 UTC. Live.twit.tv.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details: http://creativecommons.org/licenses/by-nc-sa/2.5/