Geneva

Description: This week we note the completion of the first virtual Black Hat and Defcon conferences. We also examine the latest academic work to emerge from the Graz University, which dramatically advances our understanding of the past few years of performance optimizing processor vulnerabilities. We look at the ransomware attack on Canon, a mishandled vBulletin vulnerability disclosure, the forthcoming support for DoH on Windows 10, and the result of Troy Hunt's yearlong quest to find a home for his much-loved "Have I Been Pwned" services. We have a bit of miscellany, some feedback, and an update on my SpinRite work. Then we examine a very interesting new technology being used to evade state-based Internet censorship known as "Geneva."

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-779.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-779-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. He's got reports of some of the hacks revealed at Black Hat and Defcon. We'll also find out why speculative execution is something that's going to plague all processors for the foreseen future. I don't know what we're going to do about that. And then an AI bot designed to find security flaws. They're turning it against the Great Firewall of China. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 779, recorded Tuesday, August 11th, 2020: Geneva.

It's time for Security Now!, the show where we cover your security, now, with this guy, now, right here, Steve Gibson. He is the guru of Security Now! and has been for near, lo, near all these 15 years. Is that next year, next week that we're going to celebrate that?

Steve Gibson: Think it is. I looked at it a couple weeks ago, and it's not on the day, but it's plus or minus one day. So I think next week, 780, is the end of our 15th year; and then the next podcast, 781, begins year 16. But this week we're at 779 for August 11th, and a really interesting topic. You wouldn't know from the name of the podcast, "Geneva," what this is about. And we're going to tease it a little bit for our listeners.

But we have a lot to talk about. We're going to note the completion of the first virtual Black Hat and Defcon conferences. And there was some news that came out of there, but on close examination nothing really stunning. Like even that satellite, the guy that had been monitoring satellite communications forever and spotted a few things that were in the clear, it's like, yeah, well, okay. If you watched any communications channel for a long time you'd see some leakage. But anyways, we'll talk about that.

We're also going to examine the latest academic work to emerge from our friends at the Graz University. They have dramatically advanced our understanding of the past few years of performance optimizing processor vulnerabilities. Basically they stepped back and realized everyone had been wrong about what they thought the problem was. They generalized it; and, as a result, have found a new set of attacks which apply to all processors - Intel, AMD, IBM, it doesn't matter. ARM, just across the board.

Leo: Wow.

Steve: We're also going to take a look at what we know about the ransomware attack on Canon. A mishandled vBulletin vulnerability disclosure which was just unconscionable, we're seeing that again. We have the forthcoming support for DoH on Windows 10. Also the result of Troy Hunt's yearlong quest to find a new home for his much-loved Have I Been Pwned services, and some interesting insight thanks to his posting about that. We've got a bit of miscellany, some feedback, and an update on my SpinRite work. Then we're going to examine a very interesting new technology being used to evade state-based Internet censorship, which is what Geneva is the acronym for. So I think a really interesting couple hours for our listeners. And our Picture of the Week is an oldie but goodie.

Leo: Oh, so it's not a hack. It's a good thing, Geneva.

Steve: Oh, yeah. Yeah, yeah.

Leo: Yeah, I think - in fact, when I saw the Picture of the Week, I thought maybe we've done this before. But maybe not.

Steve: We have. But not for a while. And still it's just...

Leo: It's a classic.

Steve: You've got to say, what is the story behind this one?

Leo: It's a classic. It would get, let's put it this way, a security scorecard of F. As low as you can get.

Steve: Well, yes. And it's reminiscent of the gate in the middle of the path, where you can see the foot traffic that just walked around the gate because, okay, again, what were they thinking?

Leo: All right. We're going to get to it. It's going to be a good show. Steve?

Steve: Okay. So our blast from the past, oldie but goodie Picture of the Week is, I mean, it looks authentic. Maybe the whole thing is a spoof. But if nothing else it's fun. It's an 8.5 x 11 sheet, apparently stuck on some wood panel, that is titled "Password Change

Signup Sheet." And then the instructions are, "If you would like to change your password, please fill out the form below, and we will change your password on the system you indicate."

So, okay, first of all, that's a problem. So this is public, apparently. It's like, you know, hung next to the coffee machine or the water cooler. And it's got the person's full name, then a column for which system. And it says Yardi, email, et cetera. And examples are email, phone, and Facebook have been put down in the column. Then the person is giving their current password and then their new password. And in the case of Jack H., who'd like his email password changed...

Leo: They're all terrible passwords.

Steve: He's not being very creative. The current password is reportedly "Password."

Leo: Okay.

Steve: And the new password is I'd like it changed please to Password2.

Leo: Really up the security with that.

Steve: Yeah. And Liz wants her phone password changed. Currently it's 89621, which suspiciously looks like maybe her zip code. But anyway, she wants it changed to something simpler, apparently, just 4281. And we've got, I don't know, Kyle. The current password looks like Scooter49$, so that showed some originality. He's wanting it to be changed to Skeeter442. And then he says, "(all upper case)."

Leo: Please, yes.

Steve: Even though he didn't write it in uppercase. Anyway, so we don't know what the hell is going on here.

Leo: I think it's a joke because you notice Sam Adams, whose password, he wants it to be "beer lover 1981." I'm thinking it's a little bit of a joke.

Steve: Okay.

Leo: But you know what, it could be real, too. This would be a security scorecard of an "F," for sure.

Steve: Well, yeah. And then what happened was Shawn decided, I don't know what it was, but he wrote "Come see me" on a Post-it note.

Leo: I think Shawn's the boss.

Steve: Signed Shawn.

Leo: Shawn's the boss, yeah.

Steve: And it may have been the Sam Adams because that's the last entry in a list of five.

Leo: After that, yeah.

Steve: So maybe Shawn thought, okay, wait, you know, maybe this was not such a good idea. Anyway, but obviously the point is, well, everybody gets the point. This is just lunacy. But anyway. If for nothing else, a fun Picture of the Week for the podcast.

This is August 11th, making it August's Patch Tuesday. And Patch Tuesdays have become so eventful - which is really not the word you want associated with patching your computer. But they have become so eventful that we now talk about them on the Tuesday following the Patch Tuesday so we can discuss the events. Woody Leonhard, who is a well-known, long-time PC personality, he writes his column for Computer World, he tweeted from @AskWoody: "MS-DEFCON 2." He said: "Patch Tuesday's tomorrow. Take a minute to make sure you have Automatic Update paused. You have to get patched sooner or later, but there's no reason to join the ranks of the unpaid beta testers." Which is pretty much the position he has taken recently. So we'll cover all the details and, now, sadly, the inevitable fallout of what happens today, next week.

And speaking of weeks, we have last week, which was August 1st through 6th was the first virtual Black Hat event. I looked through all of the disclosures. Many of them are, sort of, of special interest, or maybe somewhat obscure. But their titles, if nothing else, are quite titillating. And I noticed that many of the issues are things we've already covered in previous weeks or in some cases months where I noted that this would then be discussed at the forthcoming Black Hat, which was last week. And indeed it was. So, and there were some interesting things. But nothing, like, really jumped out at me.

I created one of my little shortcuts because it takes a bit of navigating to get there. So if anyone is curious just to scan the list of presentations, grc.sc/bh. Again, grc.sc for shortcut, /bh. That will take you to the recent Black Hat presentation list. And if you click on any one presentation, you might see something that you want more information about. You'll find a summary of the presentation at some length, and then typically links to the presentation's slides, often an accompanied detailed whitepaper that can be downloaded. And that's the kind of thing we normally discuss in detail sometimes months in advance of the actual presentation. And in some cases even an accompanying software tool that can be downloaded to demonstrate whatever the presenter has, typically hosted over on GitHub.

So anyway, a little bit of a self-serve Black Hat because it was virtual. Everything is online. And nothing, again, nothing really jumped out at me. I looked through everything. And it may be that we end up covering something that ends up to be more than it looks like.

Leo: You're going to cover Achilles; right? I mean, that seemed like a serious - maybe not?

Steve: I'm not even sure what you're referring to.

Leo: That's a vulnerability that affects 40% of all Android phones.

Steve: Ah, I did see that and did not dig into it.

Leo: Yeah, I think that one's maybe fairly serious.

Steve: That was the Snapdragon vulnerability?

Leo: Yeah, big Qualcomm Snapdragon vulnerability. And, you know, a billion phones. The problem is most of those phones are older, so they won't be patched.

Steve: Right. They're never going to get patched.

Leo: There is a patch, but they're not going to get it.

Steve: Yeah. Which unfortunately is the refrain these days. Which, as we know, for our listeners, for what it's worth, we talk about absolutely looking at the patching reputation of anyone from whom you purchase a smartphone because it's a computer in your pocket, just as much as our desktops and laptops and workstations are computers.

Leo: Yeah, I mean, Samsung just announced its new Samsung Galaxy Note 20 and said, hey, good news, three years of updates. And I'm thinking, that's all? But that's pretty typical in the phone space.

Steve: And that's a problem because on one hand it would be nice if they were committing to updates for the service life of the device. On the other hand, you can understand, much like Windows, or like Microsoft, that finally decided to kill off, I mean, explicitly kill off support for previous versions completely. Although, you know, you have to look at what has happened to Windows 10. And now it's just a continuous rolling new version of Windows 10. So it's not clear that anything changed there. But what has changed is they are forcing you to move forward, rather than allowing you to stay with an older version for an indeterminate length of time.

So the Graz University gang are at it again. Their most recent paper is titled "Speculative Dereferencing of Registers." And, they said: "Reviving Foreshadow." I've got a link to the paper in the show notes, but I'll summarize it for people because there's nothing we as individuals are able to do about this. In their paper they clearly demonstrate that, without a carefully conducted, purely academic understanding of the impact of modern processor design optimizations, which hadn't happened until now, it's not possible to make those designs truly secure. They show that the actual fundamental root cause underlying many of the previously disclosed speculative execution attacks such as Meltdown and Foreshadow were or was misattributed to the effect of prefetching. And this misapprehension resulted in the hardware vendors effectively rushing out and releasing incomplete mitigations and countermeasures for the problem. That is, they didn't actually fix it.

So this paper, which is now public, shows that microarchitectural attacks, and we know that to mean attacks affecting the microarchitecture like the microcode and the things it's doing behind the instruction set, were actually caused by speculative dereferencing of user space registers in the kernel, and that it doesn't only impact the most recent Intel CPUs, which as we know have received most of the attention in the last couple years, including those from Intel carrying the latest hardware mitigations, meaning it still hasn't been fixed, but also several modern processors from ARM, IBM, and AMD which were previously believed to be immune. So, I mean, this is sweeping impact. So in other words, by developing, as they have, a true understanding of the problem, they've come up with a new approach to attacking the security guarantees provided by all modern processors. All modern processors.

Leo: You know who I bet's paying close attention to this is Apple, since they're in the process of developing new silicon based on the ARM architecture. Just because it's based on it doesn't mean it can do the same thing. I would think they would want to kind of eliminate these speculative execution pipelines and things. I don't know.

Steve: Yes and no. I'm going to come up with a possibility here at the end of this because when you...

Leo: Yes, I want your prescription for what processors should do going forward.

Steve: Exactly. So what they said in their abstract is not something we need to deeply understand, but it will give us all a better sense for this. They said in the abstract of the paper: "Since 2016, multiple microarchitectural attacks have exploited an effect that is attributed to prefetching. These works observe that certain user-space operations can fetch kernel addresses into the cache. Fetching user-inaccessible data into the cache enables KASLR" - we know that that's Kernel Address Space Layout Randomization - "breaks and assists various Meltdown-type attacks, especially Foreshadow. In this paper we provide a systematic analysis of the root cause of this prefetching effect.

"While we confirm the empirical results of previous papers, we show that the attribution to a prefetching mechanism is fundamentally incorrect in all previous papers describing or exploiting this effect. In particular, neither the prefetch instruction nor other user-space instructions actually prefetch kernel addresses into the cache, leading to incorrect conclusions and ineffectiveness of proposed defenses. The effect exploited in all these papers is, in fact, caused by speculative dereferencing of user-space registers in the kernel. Hence, mitigation techniques such as KAISER" - which is the solution in Linux - "do not eliminate this leakage as previously believed.

"Beyond our thorough analysis of these previous works, we also demonstrate new attacks enabled by understanding the root cause, namely an address translation attack in more restricted contexts, direct leakage of register values in certain scenarios, and the first end-to-end Foreshadow exploit targeting non-L1 cache data. The latter is effective even with the recommended Foreshadow mitigations enabled, and thus revives the Foreshadow attack." And they conclude: "We demonstrate that these dereferencing effects exist even on the most recent Intel CPUs with the latest hardware mitigations, and on CPUs previously believed to be unaffected, i.e., ARM, IBM, and AMD CPUs."

So what does this translate for us in the real world? The researchers established a "cache-based covert channel," as they described it, that was able - they demonstrated it - able to exfiltrate data from a process running on an Intel Core i7 6500 CPU to their

stealth process, achieving a transmission rate of 10 bits per second and relaying a total of 128 bytes from the sender process to the receiver. Now, as we know, we're not looking at attacks any longer that export the whole process space. That's not necessary because typically what you want are keys. 128 bytes, which they were able to pull out at 10 bits per second, is the length of four 256-bit cryptographic secrets. So if you can steal the keys, you've pretty much got the kingdom.

They also worked out a means of leaking the register contents from an Intel SGX enclave, extracting a 32-bit secret value stored in the enclave in a 64-bit register within 15 minutes. And to give their research a bit more bite, they also demonstrated some of their attacks can be performed at a distance using JavaScript combined with WebAssembly.

So this is mostly to get the attention of the people for whom it matters, which are the chip vendors and the OS vendors. There's nothing that we as end users need to or can do about any of this. In this we're all spectators. The only full and true solution requires our processors to even further back away from their dearly beloved performance enhancements. And Leo, to your point, it may be that someday we'll be given a choice between performance...

Leo: And security.

Steve: ...and reduced security, yes. Or full security with a performance compromise. And you know, as we've talked about this, when we talked about Spectre and Meltdown and so forth, and how, well, first of all, this is all theoretical. There hasn't been a single attack that's ever been demonstrated practically or in the wild. It's just been security researchers saying, yeah, but we could. And the environment matters. I would argue it would make sense for unshared personal PCs to opt for performance, whereas cloud-based systems where they have a shared infrastructure, and they're also super beefy machines, they would probably have to take the enforced security route. And this might be something you set in hardware, like a jumper on your motherboard. So it's not even a BIOS thing that could be changed because that's too soft.

Maybe it's a different processor, where Intel says, well, we'll sell you the speedy one that's got all of the triple scoop enhancements that we know how to do. But unfortunately it's got Spectre, it's got Meltdown, it's got Foreshadow. It's vulnerable to all this. But who cares? If it's your computer, your different processes are the only things running on it. And if something gets into your computer, then doing some really bleeding-edge Spectre thing is not what any malware's going to do. It's going to encrypt your hard drive, or it's going to do something sort of blatantly nefarious, not try to leverage some very subtle cross-thread or cross-core cache leakage. It's just not going to.

But in the meantime, since you're not worrying about all that, you get top-line performance from this processor. Or if you do have an environment where security is more important than performance, and by that I mean this kind of like, okay, it's probably never ever going to bite you kind of security, except you have to be able to say we've chosen the Intel secure core system, yes, at two-thirds the performance, because we've had to disable all this stuff that the academicians have demonstrated could provide leakage. But it's secure. Certainly there's a market for that.

So, I mean, I can see it having, like, it going both ways. What this comes down to clearly is that in order to optimize the operation of our systems, we must have caching. And we must speculate. We've got to load pipelines with both outcomes of a branch so that, regardless of which way the branch goes, we're already starting to execute down both

paths because that's the way we get state-of-the-art performance is it's not the road less taken, it's both roads taken, and then we decide which outcome we want as soon as we figure out what the result of an earlier branch was.

We have, I mean, insane as that technology is, Intel has implemented it. And that's the way we got the performance that has spoiled us. But that inherently means that there will be traces of those decisions left behind when execution is suddenly changed to a different process. And that's the way we do multiprocessing these days. Now, maybe that could change so that instead of it just being a timer event, which ticks, and then suddenly a thread context is changed, maybe the solution if for the OS and the processor to agree on when it is safe to make a context switch.

So maybe we need to rethink the way we share a set of cores among a much larger set of processes and OS instances or VMs. Maybe that's the route we'll take, some sort of a flushing or a safe-to-switch boundary. Or it may be that you use lower performance systems if you really, really want to say, you know, even the academicians are finally happy, even if we need to add a few more processors to the system in order to get the same level of performance. Anyway, the guys at Graz did it again. Looks like this time they really cut it down to the bone, understood the nature of the problem. And this really leaves us with a choice.

Leo: It's inherent in all modern architectures that we have to do prefetching? Or is it, I mean, is it conceivable that there could be optimizations that aren't susceptible to this kind of thing?

Steve: So the way Intel does the Intel architecture and all the others do their paging, this mapping between virtual memory and physical memory, it really is insane. It's like four layers of indirection with these paging tables, which technically are in RAM. But if you actually had to go get them from RAM, I mean, it would take you a year to boot your machine. I mean, because you can't be going to get them from RAM. So they have to bring them into the hardware in order to make the lookup fast. And even then they cache the result of the four-way page table lookup, so they only have to do that when they don't have a result from the most, you know, a recently immediately previous lookup because that's the only way to make this system work. Even that can be a source of compromise.

Leo: So what about Optane, or some sort of faster RAM, or RAM that's closer to the CPU, or, I mean...

Steve: Yes. And that's what's interesting. Remember that all of this is the consequence of the fact that DRAM, which is based on a large sheet of capacitors, it has proven absolutely intractable for getting any speed increase. Our processors just shot head of where DRAM remains.

Leo: Interesting.

Steve: But there is that - what is that...

Leo: Crossfire?

Steve: The Intel XPoint?

Leo: Cross, yeah, that's Optane now. That's the brand name.

Steve: Yeah. The Optane memory is interesting because, although there's some concern about fatigue, it is at least dramatically faster. And so if Optane gets built onto the chip, then maybe this can push this caching problem further down. But also, even up at the higher level, technically speculation is caching; right? Because we're pre-executing multiple paths and caching the outcome so that once a branch whose outcome we've been waiting for is determined, the processor's already gone down both paths and has got our results waiting for us.

So there are, I mean, it just turns out to be like all the clever things that designers have come up with leaves a trace down in the microarchitecture. And the security researchers said, you know, you just can't do that. We can sense that, and that allows us to determine what another process was doing when context switched to us. So I think some level of rethinking, maybe, of the way processes and/or virtual machines share the processor is going to be necessary.

Leo: I bet you there's people thinking about that.

Steve: Remember one of those attacks, there was a really cool - one of the Rowhammer attacks used the fact that you could have many more instances of Linux sharing a single hardware platform because they were all running Linux, and that meant they all had a copy of the same OS, which meant that the virtual machine manager was clever enough to point all of these separate VMs to the same physical memory.

Leo: Right.

Steve: Instead of them each actually having what looks like a copy of their own physical memory. In other words, they were doing page table collapsing where they were actually all sharing this, I mean, it's very clever and very cool.

Leo: It's my understanding that's roughly how containerization works, as well. And you know the world is moving towards containerization like Docker. That's one of the benefits of it is that, you know, you can share the OS space.

Steve: Right.

Leo: So, boy, I am - we clearly went down a wrong path. Do you think these guys are right?

Steve: It was nice for a while, though.

Leo: These guys - yeah, it was good while it lasted. It's probably prudent at this point to reiterate that this only affects people with shared processes on the same

processor, multiple different users with the same - because one of them could be malicious and could see in theory what you're doing in your process. So that's why it's a server issue more, or a virtual private server issue more than it is an individual computing issue. But, you know, honestly, the future in my opinion is that we're going to have - we won't need such processors on our desktop because we'll have server processors. That's what this cloud gaming is. And this could be a big dent in that possible future. It's very interesting.

Steve: Yeah.

Leo: There's got to be a way around it. They don't have to do speculative execution. Right? Surely there's somebody, some bright guy is going to come along and say, oh, well, what if you did this? And then in 10 years we'll be talking about the next thing.

Steve: You know, there is the possibility - it all comes down to the problem with the go-to instruction. And if you changed things around so that you had the come-from, then that would really...

Leo: I told you you guys should have used Lisp. I've been telling you this. You and your go-tos. It's really an interesting conundrum. And this is - these guys are reliable; right? These are the guys who came up with the initial research.

Steve: Oh, yeah. Oh, yeah, yeah, yeah, yeah. They are, you know, they just they made their thesis advisors very happy with this one.

Leo: Yes, yes. Wow. See, that's a big story. That really is a big story, yeah. Fascinating.

Steve: Yeah. And the fact that it's not some mistake Intel made.

Leo: No, everybody made it.

Steve: It's fundamental.

Leo: Yeah. Well, and they didn't make a mistake. It was a choice that turns out to have unfortunate consequences.

Steve: Yeah. And this goes deep. Remember that we talked about how someone a while back did comment that, well, you know, if you do the speculation, then there could be some cross-process leakage of information.

Leo: It was like '67 or something, or '76 this paper came out.

Steve: Right, right.

Leo: And it was like the year before they started doing it. So it was kind of known there's a potential here.

Steve: Yeah.

Leo: Maybe we just didn't have the tools to achieve it or something. I don't know. I don't know. Back to the drawing board. And it ain't gonna be Intel this time.

Steve: We certainly do know that there has been an arms race for processor performance.

Leo: Right.

Steve: And so it's understandable that without this being anyone's fault, really, the engineers did say, oh, wow, you know, we've got so much silicon now, we don't have to wait to see which branch the code takes. We'll just do them both because, you know, why not? And someone said, "Great idea. Go." And then so that seemed to be good. You know? It worked. It executed the instructions correctly, and everybody was happy with it for, you know, quite a while.

Leo: Yeah.

Steve: And then someone said, wait a minute. The music ended, and there were like a lot of people standing and not...

Leo: Hey, where's the chairs? Where did the chairs go?

Steve: Yeah.

Leo: It was 1995 somebody wrote this paper saying, hey, you could have a problem with this. "The Intel x86 Processor Architecture: Pitfalls for Secure Systems."

Steve: Yup.

Leo: 1995 IEEE Symposium. Warned against a covert timing channel in the CPU cache and translation lookaside buffer. This was under the auspices of the NSA, which said, "Shhh, ixnay on the security issue. Let it be that we know what no one else does. What do you say?" Wow. The NSA. Wow. It's been going on for a while.

Steve: So one quickie, and then we'll take our second break because this is not big. We can add Canon, you know, Canon Corporation, to the growing list of large high-profile

companies who have a lot to lose, who have recently had their security compromised by a quite public ransomware attack.

Leo: Gosh darn it.

Steve: BleepingComputer was all over this one. And as they did with Garmin, they solicited and were able to obtain some previously unpublished internal memos in advance of any Canon official disclosure. As we know, Canon's many services dropped offline on Thursday, July 30th, and remained down until they began returning to use six days later, a week ago, last Tuesday the 4th. So this is a copy of what BleepingComputer got. It's titled "Internal Message from the Crisis Management Committee." Canon USA, Inc. and its subsidiaries understand the importance of maintaining the operational integrity and security of our systems. Access to some Canon systems is currently unavailable as a result of a ransomware security incident we recently discovered. This is unrelated to the recent issue which affected image.canon.

Leo: Oh. I thought it was the same thing.

Steve: I know.

Leo: Oh.

Steve: That's what they're saying. It's odd that the timing was what it was.

Leo: Oh, wow. Yeah, people were furious about image.canon. They lost a lot of originals.

Steve: Yes. They said: "We immediately implemented our response protocols and began an investigation. Cybersecurity experts who have worked with other companies that have had similar issues have been engaged. We are working quickly to address the issue and to restore operations. As updates are available, we will do our best to communicate via email and ENS," whatever that is. "We appreciate your patience as we work through this incident."

Leo: If I were Canon, I would blame the image.canon loss on this. Because what they're saying is, oh, no, it had nothing to do with that. We just made a mistake in the code. And that's why we lost all your images.

Steve: Ooh. Yeah.

Leo: I would have said, oh, yeah, could have. It could be the ransomware. Definitely.

Steve: It's the bad guy, yeah, it's the bad guys.

Leo: And by the way, Canon said today, you ain't getting those images back. They're gone.

Steve: Ooh, boy.

Leo: Folks, I hope none of you used it as your only backup. Remember what Leo says, and Steve, too. One copy of anything is not a backup.

Steve: That was 10GB of free image upload and storage.

Leo: Gone.

Steve: Yeah. So we don't know anything about the way they got back online, how it was resolved, whether a ransom was paid or not, blah blah blah. BleepingComputer did identify this, however, as the Maze ransomware, so different than the one that zapped Garmin. But still, you know, this has got to be the issue that keeps companies and IT people up at night. I mean, at this point, how many times have we said it? I understand the challenge when, as Garmin did, you have 30,000 workstations, and this thing rifles through them and gets to them. That's going to be tough to bring back online. But unfortunately, that's the price of playing the game these days. Wow. Leo, we will talk about a very sad event and disclosure with vBulletin after our second break.

Leo: Wow. What a world we live in.

Steve: Yeah.

Leo: What was it they - now they're saying Garmin paid $10 million to get their data back.

Steve: Well, that was the asking price. I don't think...

Leo: Think they negotiated it down?

Steve: They could have. There was a similar incident where the 10 million was asking, and they ended up paying 4.5, I think it was. So still, you know, for a company like Garmin, if they really didn't have backups? And of course the other problem now is exfiltration, you know, the embarrassment, because that's now, you know, it's okay, you may decide not to pay us, but we're going to just post all of your secrets.

Leo: That's why they get on the system, and they exfiltrate for three months, and then they ransomware.

Steve: Yup.

Leo: That's the worst thing that can happen.

Steve: Okay. So this story begins just two days ago, on Sunday, August 9th, with a posting by a security researcher who goes by the name Amir Etemadieh, and his Twitter handle is @zenofex. He maintains a blog at "exploitee.rs," so sort of exploiters. And to give you some feel for his approach, his recent postings have been titled "Rooting the Fire TV Cube and Pendant with FireFU," "All Your Things Are Belong to Us," "Hacking the Western Digital My Cloud NAS" (Network Attached Storage), "Re-Hacking The Samsung SmartCam."

So this guy clearly likes poking at things and documenting his discoveries. He's also currently looking for a job. He recently posted a pinned tweet addressed to Twitter, stating on July 6th: "Hey Twitter. I'm actively looking for a job. An ideal role would include full-time exploit dev," he says, "(remote) with great benefits. I have crashes, fuzzers, and a good zero-day. If you have an opening that you think I may be a good fit for, contact me, zenofex@" and then http://zenofex.com.

Now, unfortunately, he followed up that job solicitation by Sunday irresponsibly posting his discovery and full details of a true zero-day exploit, a remote code execution exploit, against vBulletin, without ever giving them a heads-up. You know, I'm not Twitter.

Leo: Oh, that's too bad.

Steve: I know. He's clearly a talented reverse engineer, and I'm sure he understands what he has just done, and somehow he doesn't care. But I would not want this person inside my company. So they say when you apply for a job these days, the content of your social media is typically scrutinized. Well, were I considering someone to hire, it wouldn't be someone who was essentially boasting about a critical remote code execution vulnerability, posting full details in multiple languages, even a slide deck about it, and saying, you know, good luck.

So this particularly explosive security bomb he titled "Exploiting vBulletin: A Tale of a Patch Fail." And he wrote, he said: "On September 23rd" - so September 23rd, 2019, right, like almost a year ago - "an undisclosed researcher released a bug which allowed for PHP remote code execution in vBulletin 5.0 through 5.4. This bug (CVE-2019-16759) was labeled a 'bugdoor' by a popular vulnerability broker because of its simplicity, and was marked with a CVSS score of 9.8" - you know, 10 is the maximum.

Leo: A 10? That's not good.

Steve: Yeah. So he writes: "Today, we're going to talk about how the patch that was supplied for the vulnerability was inadequate in blocking exploitation, show how to bypass the resulting fix, and release a bash one-liner resulting in remote code execution in the latest vBulletin software." He says: "The vulnerability mentioned above was formally labeled CVE-2019-16759, and a patch was issued," blah blah blah. So that kind of repeats things. He says: "Although the patch was provided in under three days, the patch seemed at the time to fix the proof of concept exploit provided." In other words, vBulletin immediately responded and had it patched in under three days.

Okay. Then Amir proceeds to take this apart, this previous work, step by step. And like Sandbox Escaper, he's clearly a skilled technician. His posting beautifully walks its reader through the way vBulletin's on-the-fly PHP authoring engine operates. And he provides a

proof of concept, and not just one. He first offers a single cURL command line and provides a bash script he promised. He also shows a Python exploit and indicates that he's in the process of pushing a public Metasploit module. He even produced and published a deck, as I mentioned, of PowerPoint slides, a link for which I have in the show notes.

He then concludes by offering what he describes as a short-term fix. He says at the end of his posting: "This fix will disable PHP widgets within your forums and may break some functionality" - he doesn't say so, but ads do break - "but will keep you safe from attacks until a patch is released by vBulletin," whom he did not inform of this. So you go to vBulletin administrator control panel, click Settings in the menu on the left, then Options in the dropdown, choose General Settings, and then click Edit Settings. Look for Disable PHP, Static HTML, and Ad Module rendering. Set to Yes. Click Save.

So what happened next? Unfortunately, vBulletin sites are easy to locate with any Google search for the phrase "Powered by vBulletin." And vBulletin is one of the more popular forum software systems. It is in use by Electronic Arts, Zynga, Sony, Pearl Jam, NASA, Steam, and many more. The attacks started almost immediately upon his publication. Jeff Moss, who's known as "The Dark Tangent," perhaps best known as the creator of the Black Hat and Defcon security conferences, tweeted: "A new vBulletin zero-day got dropped yesterday by @Zenofex that revealed the CVE-2019-16759 patch was incomplete. Within three hours, forum.defcon.org was attacked, but we were ready for it," whatever that means. He says: "Disable PHP rendering to protect yourself until patched!"

Amir's reply to Jeff was "Thanks for the mention," though I'm sure Jeff would have preferred not to have his Defcon.org site attacked, if this had been done responsibly. And around the same time Amir tweeted: "Looks like the @vBulletin forums are currently down while they patch from the RCE vulnerability release. Hopefully that means customers will also see a working patch soon."

Yeah, gee. Or instead of being attacked by an irresponsibly disclosed zero-day, customers would probably have much preferred that he disclosed this quietly to vBulletin, who would certainly have taken it very seriously. Last September's publication of the original zero-day, the fix for which I mentioned, or the fix for which this is a workaround, triggered a massive wave of vBulletin hacks which resulted in many companies disclosing security breaches through the months that followed. Since Amir did disclose this vulnerability, it's clear that he didn't want to use it himself. But the disconnect here seems to be that he won't mind at all if he enables, directly enables others to wreak havoc in his name.

And sure enough, yesterday we have a thread on the vBulletin support forum, the first one I found this morning. "I received the scan results on my server that shows a URL - known exploit, fingerprint match, PHP shell exploit - indicating I have some type of issue. It was recommended that I replace the web.php file." Which, by the way, is not an official file on vBulletin. That's something that somebody installed and then ran. "That is fine, but I don't know what this file does or where another copy of it would be."

Yesterday at 7:27 a.m.: "My site was also compromised this morning at 6:35 a.m. GMT-7. The file is named 1.php and was placed in the root of the site. Windows Defender identified the exploit as Backdoor:PHP/Shell.Q. I've zipped up the file and attached here as Backdoor."

Yesterday, 11:15 a.m.: "I got hacked at noon CST today. Was on 5.6.1 Patch Level 1. Have upgraded to 5.6.2. Can provide HTTP access and error logs to Wayne Luke if you guys need help determining the attack vector." Wayne works for Support at vBulletin.

And at 11:25 a.m.: "We know the vector, and a patch is being developed. Hopefully I can say more soon. The only workaround I can say at this time is to make your vBulletin directory read-only (chmod 444)." And of course that relates to we were talking about Linux directory privileges. Turns out that remediation was found not to work because it blocked some access that was needed to the .htaccess file.

Yesterday at 11:44 a.m.: "I found the files hax.php at these locations," and then two URLs. Yesterday at 12:20 p.m.: "My 5.6.1 site was hit, and I cannot access either URL. I have 444'd it." He says: "My 5.6.2 sites appear unaffected." Yesterday at 12:57 p.m.: "Hi, is this the best place to be kept up to date about this issue? Our 5.6.2 forum was hit as well today. Is there a CVE number to track the issues yet?" And then, finally, yesterday at 2:06 p.m.: "Patches are available," and a link to the patches.

So this is not just theoretical. When you loudly disclose a simple-to-execute remote code execution vulnerability like this, it hurts people. I mean, it messes things up for people. And so I'm at a loss to understand, like, what's in the mind of a person, especially who says, "Hey, Twitter, how would you like to hire me?" Right. Like Twitter's in the mood for that right now, after what they've just gone through. So needless to say, anyone using v5 of vBulletin, that needs to have been patched last Sunday, two days ago. Patch it now, if it hasn't been.

Leo: Oh, geez. Wow.

Steve: And to the common refrain that it's crucial to keep lines of communication open to the vendors of the software you're using, in other words, you'd like to have received this notice from vBulletin yesterday, I'll now add that this reality suggests that some level of compromise needs to be factored into any security-conscious organization. It's similar to the fact that intrusion detection has now become an accepted part of an overall security posture. Detecting an intrusion by preparing for it inherently acknowledges that there's going to be or might be an intruder to detect. And that's never good. But if there's going to be an intruder to detect, it's far better that it be detected than not.

We know that the proper way to think about security is that it is inherently porous. I hate the truth of that, but we see if over and over and over. You know, the adjective "porous" is the one I've adopted and used on the podcast because I think it perfectly says that the more pressure you put against security, the more likely it is that something will leak through. You know, it's not absolute. The sobering truth is that this means that systems must be made less brittle. By that I mean that the event of a security breach should bend, but not break, an organization's security. A practical example of this might be to recognize that PHP-based forum software has grown quite complex. And we all know what that means. With complexity comes risk.

So to mitigate the risk, software like PHP-based forums should be run on their own hardware behind a firewall such that even gaining root access on that machine, that firewall cannot be bypassed. So not just a firewall running on the machine, but a firewall outside of that machine. The typical forum only needs to have access to an organization's DNS and email servers. I would argue that it should have its own local SQL database and not be reaching outside to the organization's shared SQL server. There's a recipe, as an example, a recipe for disaster. Containment in the event of a breach is the whole point.

And since DNS and email services exist at fixed addresses and only need a couple of ports, and they're comparatively benign if they've been set up right, a forum hosting machine's network connection should be placed behind a "deny any" external firewall that only makes exceptions to that "deny any" policy for a very few ports. After all, this is the reason we call a firewall a firewall. If anything catches fire and explodes on that

machine, which in this case might be inevitable if it's hosting complex PHP-based forum software, at least the damage will be contained.

So in other words, think about what if, you know, pose "what if" scenarios and then ask where you can impose boundaries of containment so that your overall corporate security posture is flexible. If something happens, okay, ouch. But it didn't let somebody get out of a PHP forum onto your network, you know, execute code, get onto your network, and then set up a much more potent advanced persistent threat by then putting files up on a shared SQL server and going from there. The world we live in today.

As for the world we don't yet live in today, DoH for Windows 10. We don't talk much about Fast Ring and Slow Ring or the Release Preview Ring because even being on the "how do I get off this rollercoaster ring" provides sufficient excitement these days on Windows 10. But those rings fascinate Paul, and especially Mary Jo, who talk about them all the time over on Windows Weekly.

And while we're on the subject of rings, I'll note that the paint is still wet on the renaming of those rings, to now refer to them as "channels." That's the official new designation. In order of decreasing excitement and increasing sanity and stability, we have the Dev Channel, the Beta Channel, and the Release Preview Channel. Rings are gone. We have channels now.

So as we know, and we talked about it in November, way back in November of 2018 Microsoft surprised us by announcing that Windows 10 would natively be getting encrypted DNS using DoH, DNS over HTTPS. Apparently not DOT, but that's the way they, I mean, so they're like going the way everybody else has. So that would mean that not just random browsers running on Windows but everything in the OS would get its DNS lookups encrypted. This will actually happen for the first time when the Dev Channel - previously known as the Dev Ring, now the Dev Channel - gets the Windows 10 Insider Preview Build 20185, which is coming soon to a Dev Channel near you, if you like to live dangerously.

And as I said before, this early access news doesn't move me at all since I'm definitely happy staying right where I am, getting actual work done over in the "Won't you just please stop fussing with it" channel. But once this does migrate to the regular, okay, we're going to get this in an update through Windows 10 for the rest of us, we'll definitely be talking about it since its configuration details will probably be interesting, and I'm sure that a bunch of our listeners will have applications where it makes sense for their DNS to stay away from prying eyes. You know, I'm on a Cox Cable connection. Cox sees my lookups, but then they all go everywhere else from Cox's servers, and I have nothing of interest to Cox, so I'm not that worried about it. But certainly I get it that a lot of people will be.

This is a really interesting story about Troy, Troy Hunt. And it's too long for me to cover in the kind of detail that I wanted to. But I'll give you enough of a taste of it. I hope to whet your appetite because I gave it another - I gave the longest part of the story another shortcut to encourage people to go find it. I titled this "Hasn't Been Pwned" because without success, it turns out that Troy Hunt has just concluded a yearlong search for someone to purchase or maybe merge with his very popular and very useful "Have I Been Pwned" site.

Have I Been Pwned is really a much bigger deal than it might seem from the outside. For example, in his posting of March this year, following 10 months of work to find a partner, he noted: "A lot happens in 11 months. I onboarded five new governments onto HIBP: Austria, Ireland, Norway, Switzerland, and Denmark," he says, "and a sixth one about to be announced any day now. I loaded 77 new data breaches comprising 1.7B records into Have I Been Pwned and signed up almost 400,000 more individual subscribers to the

service." Holy crap. He says: "I built and launched the authenticated API and payment process. I really should have done this earlier. I'm so happy with it."

So, yeah, there's a lot there. And as I said, he posted the story of his 11-month quest back in March. It is really and truly a fantastic read. He describes the insanity of endless meetings and notes the processes' bizarre similarity to many of our much-loved episodes of "Silicon Valley." And he, like, you know, really talks about it in some detail. There were hundreds of initial candidates which he went through a screening process and a successive weaning process over with NDAs and due diligences and all kinds of stuff. And if we didn't have way too much else to get to, I would read it into the podcast. But since I really believe it's worthwhile, I've given it, as I said, a GRC shortcut, grc.sc/troy. You really should check it out if you have a chance, grc.sc/troy. I think all of our listeners would find it really interesting.

So that was as of March of this year. Troy began the wrap-up of that post by writing: "So, what's next for HIBP and for me?" He says: "To be honest, I need time to recover. What I've explained in this post will never adequately illustrate just how stressful this process was. I need some time where I'm not waking up dreading how much work will have landed in my inbox overnight. I need some time to write more code and more blog posts, two things that remain my passion but had to take a back seat during this process. I'll still keep running Have I Been Pwned as I always have, but I need the head space to get my energy levels back up and plan the next phase."

He says: "I've almost entirely cleared my calendar for the next few months" - and consider that this was in March of this year, so we know what was going on in March - "to give me that much-needed time out. And with coronavirus causing a heap of conferences to be canceled and travel plans to be disrupted, it's probably not a bad time to stay home anyway."

And that brings us to today, and Troy's posting just last Friday. I have a link to the most recent blog posting. And he starts out: "Let me just cut straight to it. I'm going to open source the Have I Been Pwned code base. The decision has been a while coming, and it took a failed M&A process" - that's what he called his Mergers & Acquisition process - "to get here, but the code will be turned over to the public for the betterment of the project and, frankly, for the betterment of everyone who uses it. Let me explain why and how." And he goes on.

And again, it's useful to read that. I've got a link in the show notes. And I'm struck again wishing I could share his detailed description of his plans for Have I Been Pwned. But at least the gist of it is clear. Have I Been Pwned will become a major and significant open source project on GitHub so that Troy is no longer carrying the entire burden of moving it forward himself. It's clear that there's huge enthusiasm for the features offered by the system. And Troy's approach has always been to be entirely open and transparent anyway. So, I mean, it just seems like a good fit and the natural next step. We'll certainly be keeping an eye on it. So the news here is Have I Been Pwned is going to GitHub and open source, and will be open to contributions and code from people other than just Troy himself.

A couple little bits of miscellany. I did have an opportunity to try Bloatbox, which was that GitHub-hosted Windows 10 debloating software that I had referred to before. And it again unimpresses. I have not yet found any app for debloating Windows 10 that does the job. And I'm still refusing to be dissuaded or, well, yeah, distracted from SpinRite. Nothing is pulling me from that. Other people can write a Windows 10 debloater. It doesn't need me. So I will keep looking for someone to point me to one, and one that I can recommend. But we still don't have it.

The Security Now! Shortcut of the Week. I've already talked about a couple shortcuts I assigned to specific things. This is the Shortcut of the Week. So it's numbered this podcast, grc.sc/779. It will take you to a page on BleepingComputer which I just sort of thought was worth sharing, mostly for the last of the seven. It's titled "Useful Registry Hacks to Optimize Your Windows 10 Experience."

The first one is disabling Bing search in Windows search. It says: "Windows Search comes with the Bing search engine integration, and it allows you to find the content on Bing when the local search fails to find anything. If you don't like Bing due to privacy or performance issues or if it's causing problems, you can disable it via the registry." And so BleepingComputer has a link to show you how. Also, disabling the Windows 10 lock screen is the second one.

Number three: Add 'open command window here' to the File Explorer context menu. I like that, actually. And as a command line user I've used that extensively in previous versions of Windows. This allows you to get it back if you, too, miss it in Windows 10.

If for some reason you wanted to display seconds on the taskbar clock, there's a way to do that. I don't need that, but maybe someone would like it. This one, though, the last one, I thought, that's interesting. Just because why not? Enable verbose mode in Windows 10. It reads: "With a tweak to the registry, you can boot your Windows 10 PC into Verbose Mode and get more detailed information, which is extremely helpful when troubleshooting problems." It explains: "This mode will display the specific step the operating system is on while booting up and shutting down Windows 10. This mode allows you to troubleshoot startup and shutdown issues to see the specific steps that are being performed when a problem occurs."

And, you know, you get that spinning rollercoaster dot thing, and it's like, I would like to know what it's doing. So I turned it on, and it tells you after you make the setting reboot. I rebooted my Windows 10 machine, my main workstation, and it showed and stalled at stopping a specific service. And sure enough, I had been wondering why my Windows 10 shutdown had been taking so long. It doesn't tell you. It immediately told me the service that it was stalled on trying to shut down. And I thought, I'm not using that anymore. So I disabled it, and now my Windows 10 shutdown went back to its normal speed.

So just a tip. I really like that one. Again, grc.sc/779 will take you to the BleepingComputer page with these links. The last one shows you that it's just a registry. You add a DWORD, set it to one, deep down in some policies somewhere. And now Windows 10 tells you what it's doing, which is, like, why not?

Also just a little bit of closing the loop. Someone who calls himself StarKiss tweeted from @StarKissedOne. He said: "Thanks for mentioning QNAP in the last Security Now!." Remember we talked about the horrible problem, 68,000 I think it was, publicly accessible QNAP servers, many of which are insecure and located in the U.S. He says: "Found that my firmware was >4 revisions out of date. Got absolutely no notifications. Love the podcast." So I'm certainly glad that our mention of this problem with QNAP did reach some people who needed it. And this, again, in this day and age, nothing more important than assuring that you have lines of open communication to the software vendors of all the stuff you use.

A quick little bit on where I am with SpinRite. We're down to the point now of finding and fixing obscure problems. Somebody had a Kingston SSD on an Acer Aspire that was causing problems. I had two SSDs, an OCZ and a something else, that were having problems. It turns out that they did not like transferring 65,536 sectors at a time. And it turns out I had discovered this seven years ago, in 2013, written adaptive code for it, and forgotten about it. Although when I discovered it, they would only go up to 65,528,

that is, eight sectors shy of the maximum. That number was haunting because I thought, I remember something about this.

Anyway, it turns out that cutting the transfer in half to 32K sectors, and thus 16MB transfers, solved the compatibility problems across the board. And because that still takes much longer than it does for me to queue up the next transfer, nothing is lost in performance, and we only gain in compatibility. There was somebody else had a drive where it was in really bad shape, and it kept basically going offline and losing its link layer connectivity. What was bizarre was it works well, or at least appears to, under Windows. So it wasn't causing any obvious problems. But the benchmark would refuse to complete because the drive was just losing its SATA link. So I ended up over a course of three iterations of test releases developing a, like, almost infinite forgivability of this drive.

And I said to the guy, I said, look, you've got to open your laptop, unplug your SSD, and plug it in again because it's got a bad connector. But don't do it until I add software to fix this in software because I want to still be able to run with it the way it is, as bad as it is. And what this means is that SpinRite will be able to notify its users when it detects problems like this because actually we've run across a few others that are having a similar problem. So it looks like it's somewhat common in SATA connections because they really do, you know, we're talking about, what is it, 6Gbps serial link. So it's a finicky link. And SpinRite will be able to diagnose link problems separate from media-related problems and tell its user, you know, we could run if we really had to, but you really ought to fix the connection to your SSD and then try again, and we'll let you know...

Leo: That would be so helpful.

Steve: ...if you've got it fixed, yeah.

Leo: So you could tell, it's a different signature if it's a bad cable than if it's something wrong with the drive?

Steve: Yeah. What actually happens is that, I mean, there's lots of error bits. And there were, like, three different error conditions. In order to squeeze the data across the serial link, there's actually - you don't put the 8-bit data on the link because the link is self-clocking. And so if you had, like, eight bits of zeroes, you wouldn't get any clocking information for too long a period of time.

Leo: Right.

Steve: So the eight bits are turned into 10 bits, and there's a fancy translation table that maps every byte into 10 bits. That guarantees that you've got a few clock interval switches that occur often enough for this thing not to lose clock lock. So that's one thing that can happen, and there are two other problems. There's also a CRC on the link. You can get a CRC link failure and then a different loss of framing. So my benchmark - because really the benchmark is sort of a test platform for the driver. So the driver keeps getting more and more sophisticated in order to deal with what we're actually finding in the field as a consequence of all...

Leo: [Crosstalk].

Steve: Exactly, a consequence of all these tests. And for a few brief hours Sunday afternoon there were no known problems with the AHCI driver code. But then yesterday a new problem with a Samsung 860 EVO on an AMD chipset appeared, and the problem appears to be the way I'm resetting the SATA port. So it turns out different chipsets want it to be done differently. So I may need to make my port resetting logic a little more heuristic than it currently is. But this is the way the sausage is made. What this iterative painstaking process will eventually produce is what everyone wants, a single AHCI driver for SpinRite that's smart enough to work on every machine and drive it will ever encounter, and to provide you with information that Windows and other OSes were hiding from you.

Leo: Yeah. Very useful.

Steve: So we're getting there.

Leo: It makes sense you have to distinguish between bad data and a bad cable. That makes perfect sense.

Steve: Yes.

Leo: Yeah, bad connection. Now it's time to talk Geneva.

Steve: So this is really interesting. China's Great Firewall tightens. A few weeks ago, China added a rule to block one of TLS 1.3's new privacy-enhancing features, ESNI (Encrypted Server Name Indication). Remember that once upon a time a single IP address could only be bound to a single web server certificate. When any client connected, a browser, the web server listening at that IP's port would respond with a certificate for the domain that was associated with that IP. This prevented the practice that is now very common and crucial, especially for IPv4, known as "multiple hosting." Multiple hosting allows multiple separate domains to all point to the same single IP. But for that to work, the client, typically a web browser, must somehow indicate from which among multiple possible domains at that IP it wishes to be served.

So an extension was added to SSL to allow the client's SSL hello handshake packet to make that specification clear. The web server answering at that IP would inspect the client's incoming SSL hello packet if it was in possession of a valid web certificate matching the domain the client was requesting in that hello packet. The web server would select that certificate for the subsequent SSL negotiation and would also remember which website to serve to the client when it then made its first URL request.

From a privacy standpoint, the trouble with this is that the so-called SNI (Server Name Indication) necessarily had to be provided to the server right upfront before the SSL connection's privacy encryption could be established. So that allowed any passive eavesdropper to easily determine to which domain this pending SSL connection was being directed. And one could imagine, if you were China running the Great Firewall, you would be interested in secure connections coming up and to which domains they are going to.

But this overt lack of privacy drove those evolving the TLS standard to work out a means, which is present now in TLS 1.3, of adding encryption to the SNI extension to create ESNI (Encrypted Server Name Indication). This is good for user privacy, but of course it's a bane for anyone having any anti-privacy agenda such as presumably the Chinese state, which we know has a manifest interest in controlling and censoring what its citizens are allowed to and able to access.

Since adoption of new standards is notoriously slow, this wasn't initially a problem for China. But as TLS 1.3 usage has continued to grow, the growing usage of ESNI as a feature of TLS 1.3 had been giving Chinese censors some headaches. I have a chart in the show notes which is encouraging. This shows, as of the beginning of 2018, so 2.5 years, where the various SSLs and TLSes are. And where we are now, the bottom two lines today, SSL 2 and SSL 3, are - looks like less than maybe 10% SSL 3, but steadily dropping. TLS 1.2 is on the rise. 1.1 is dropping. 1.0 is dropping. And 1.3 is also on the rise. So looks like TLS is almost at 100% support. And TLS 1.3 is currently at nearly one third, 31.7%. So definitely on the rise. And that means that nearly a third of connections with clients supporting it at the client end and servers supporting it at the server end would be blinding the Chinese firewall to who they're connecting to.

And as I noted at the top, it turns out that China has reacted just recently, two weeks ago, with a bit of a heavy hand, by simply blocking any TLS 1.3 connections which use ESNI. Other HTTPS traffic is still allowed through the Great Firewall, if it uses an older version of TLS or non-encrypted SNI. And it's unclear what China expects to have happen. TLS 1.3 and ESNI are here to stay, but perhaps not in China. Maybe browsers will add a feature to turn off encrypted ESNI and just fall back to SNI, which would still give you the other benefits of 1.3, but allow China to eavesdrop.

I found an entry in the IETF list archive dated from just a couple weeks ago, July 30th. It was titled in brackets "[TLS]" and then it said "Possible blocking of encrypted SNI extension in China." And the posting said: "The Great Firewall of China may have identified and blocked Cloudflare's ESNI implementation." The poster said: "I have found that when using a TLS client hello with ESNI extension to connect to server behind Cloudflare's CDN, the connection will be cut off after the whole TLS handshake is done. And then that IP address will be blocked at the TCP level for several minutes."

Some additional probing of the phenomenon revealed that the detector is not merely matching on the lack of plaintext SNI. It is specifically looking for the ESNI extension identified by the identifier that's in hex, so 0xFFCE. Also the encrypted server name extension must be syntactically correct. So it's actually parsing the extension. This poster wrote: "The detector is not just looking for the byte pattern FFCE." And of course we know that's good since no one wants false positive triggering of the Great Firewall's block on this. Actually, no one wants any triggering of the Great Firewall's block. He said: "Once an ESNI-containing client hello is detected, the firewall drops packets in the client-to-server direction for between 120 to 180 seconds," so one or two minutes. And he says the detector, that is, this block, is running on all TCP ports, not just 443.

And it's already known that most functions of the Great Firewall work bidirectionally, and this new ESNI detection and blocking does, too. Sending an ESNI-containing client hello from outside China to a server inside also results in temporary blocking. And this of course is convenient since it allows for external experimentation probing with inbound connection packets. And this brings us to GENEVA, which is the acronym for GENetic EVAsion.

A web browser's access to the domain https://censorship.ai redirects its visitor to https://geneva.cs.umd.edu, umd.edu being the University of Maryland. "CS" obviously is comp sci, and Geneva is the project within the computer sciences department at University of Maryland. It is an experimental genetic algorithm that attempts to discover,

and I should say successfully, but we'll just say "attempts to discover," how to evade censorship by manipulating the packet stream on one end of the connection to confuse the censor.

Geneva consists of two components, its genetic algorithm and its strategy engine. The strategy engine runs a given censorship evasion strategy using active network traffic. The genetic algorithm is the learning component that evolves new strategies using the engine against a given censor. Being a genetic algorithm inspired by biology, and just as biological systems compose simple building blocks - in the case of our DNA A, T, C, and G - Geneva generates new algorithms by compositing just a few simple ways of manipulating packets. Specifically, it can duplicate, tamper, drop, or fragment packets. Geneva composes these individual actions into action trees. Action trees have a trigger. This controls which packets the action trees act upon, and together these action trees form censorship evasion strategies.

Geneva initially creates many random individual strategies and runs each of them against real censors. Based on how successful they are and other factors, it assigns a numerical fitness to each individual strategy. The most fit survive from one generation to the next, and Geneva mutates and mates strategies to create new ones. The key to Geneva's success is its fitness function, which encourages the genetic algorithm to explore the space of strategies that does not damage the underlying TCP connection because of course we want connections to still work. Over many successive generations, if Geneva discovers a strategy that defeats censorship, the fitness function encourages the genetic algorithm to refine and simplify the strategy.

So one way to think of this is to compare it with fuzzing, which we've talked about often. In fuzzing, we generate entirely random junk within the limits of the target API's parameters, and we just throw it at the target to see what happens. So Geneva's a bit like that, but it adds memory and much more goal direction choosing for successive strategies. And unlike previous work of this sort, censorship evasion strategies are not manually created. Geneva discovers them automatically. Once Geneva has discovered a means of evading censorship, only then is that means analyzed manually to learn more about how the censor operates.

In this case, the work has led to interesting insights like how China's Great Firewall synchronizes its state, or how it processes packets. Sometimes the analysis concludes that Geneva has discovered a bug in the censor. An example is the segmentation species Geneva discovered in China against the Great Firewall by segmenting the forbidden request twice at specific indices. Though the Great Firewall can normally reassemble segments, by segmenting twice at these specific bounds, the sometimes less than Great Firewall can be tricked into ignoring the request to evade censorship.

As it turns out, this whole scheme actually works. Geneva has been deployed against real-world censors in China, India, Iran, and Kazakhstan. It has independently discovered dozens of strategies to defeat their censorship and found previously unknown bugs in censors. And what's more, all of these strategies and Geneva's strategy engine are open source on GitHub. And four days ago the ESNI plugin was added to the repository. I have a link to the repository in the show notes.

In the case of China's ESNI-triggered blocking, they trained Geneva over the span of two days, 48 hours, both client-side and server-side. They discovered six strategies to defeat the ESNI censorship, four which worked from the server's end, and six (including those four) which worked from the client. And they are wacky stuff that wouldn't ever occur to someone; okay? I have the first two, just because they were really interesting.

Strategy one that was discovered was the "Triple SYN." It operates by initiating the TCP three-way handshake with three SYN packets such that the sequence number of the third

SYN is corrupted. Who would ever think of trying that? This strategy performs what they called a "desynchronization attack" against the Great Firewall. The Great Firewall synchronizes on the correct sequence number, so it misses the ESNI request. It can also be applied on the server side. This means that citizens inside China could successfully connect to secure and private services operating outside China, if those services added this wacky triple SYN hack to the server. And they might only do this for client IPs originating from China.

The second example of a different strategy they called the "Four-Byte Segmentation." The client sends the ESNI request across two TCP segments such that the first TCP segment is less than or equal to four bytes. This is not the first time Geneva has discovered segmentation strategies, but it's surprising this strategy works in China. The Great Firewall has been famous for its ability to reassemble TCP segments for almost a decade. The TLS header is five bytes long; so by segmenting specifically the TLS header across multiple packets, they hypothesize that this breaks the Great Firewall's ability to protocol fingerprint ESNI packets as being TLS.

This has interesting implications for how the Great Firewall fingerprints connections. It suggests the component of the Great Firewall that performs connection fingerprinting cannot reassemble TCP segments for all protocols. And they decided that this theory was supported by other segmentation-based strategies identified by Geneva in the past. So like the first strategy, this can also be triggered server-side because, by reducing the TCP window size during the three-way handshake, the server can force the client to segment their own request, which is actually a very clever hack.

And their report on how this all works goes on like that. I have a link documenting the entire suite of censorship bypass strategies in the show notes for anyone who's interested. And of course it's true. This is a cat-and-mouse game, very much like the virus-and-antivirus game. China could, and eventually I'm sure will, respond with fixes for the problems that Geneva has found, especially since the entire effort is open and public. It's on GitHub. And maybe after that, additional ways or new ways will be found around filtering.

But I think the larger implication here is for the application of this sort of AI-ish directed discovery being applied to software vulnerabilities, not just communications vulnerabilities. I drew the analogy earlier to fuzzing. We might call random fuzzing "dumb fuzzing," which begs the question, could we make fuzzing much smarter and more effective by adding a layer of goal-directed AI to the process? Which I guess would bring us smart fuzzing, or maybe "super fuzz." And we might get more security as a result. But it does look like we're beginning to see more and more AI-style concepts being employed to security. And that's all for the good.

Leo: It's kind of interesting. So basically it picks its strategies at random.

Steve: Yup.

Leo: And just keeps trying and trying and trying till something works.

Steve: Yes, exactly. And then it finds something, and then it tweaks it to see if it works better or worse, to see if it's able to improve the way it works. And it may mate that with some other things, and it might get something even better. And it may mutate them to see if it gets something better. So, I mean, it is doing sort of a success-directed walk.

Leo: It's like a breveWalker.

Steve: Exactly.

Leo: It learns how to attack and gets better.

Steve: Yup.

Leo: You know, honestly, using this to attack the Great Firewall of China is going to have, I think, limited success. Mostly what they're doing is helping patch it, frankly.

Steve: Yes.

Leo: Mostly what they're doing is pen testing on behalf of the Chinese government because...

Steve: And nothing prevents, since this is on GitHub, nothing prevents the Chinese government...

Leo: Doing it themselves.

Steve: ...running it themselves and looking at the things it finds.

Leo: Right. Actually, what really is intriguing about this in the long run is imagine as these get smarter and smarter, these are little agents, and they're - it's completely autonomous, Steve? Or semi-autonomous?

Steve: No, it's autonomous.

Leo: It's completely autonomous.

Steve: Yup.

Leo: I think this is a terrible idea. I can just imagine these things going around, looking for vulnerabilities all the time.

Steve: Yeah.

Leo: How do you stop it?

Steve: Yeah. You can imagine, had you aimed this at port 3389, the famous Microsoft RDP port, it would have just poked at it until it found a way in and then gone, hey, found a way in.

Leo: Found a way in. The good news, it doesn't have a payload. It doesn't do anything. Does it report back? Who does it report to?

Steve: Yeah. If it were a malicious agent, definitely it would report.

Leo: You would say, hey, if you find something, let me know.

Steve: It would phone home and say, hey, I found a gene that worked.

Leo: I think we're going to look back on this and say, shoot. This may be, I mean, it was going to happen. There's nothing you can do to stop it. I'm not convinced this is, first of all, it's not going to, honestly, it's not going to take ESNI down because it's just going to find the flaws, and they'll get patched. It's going to make it better, not worse.

Steve: Yup.

Leo: And in the long run this could be a genetic experiment gone wild. All right. I'm not going to get sci-fi.

Steve: I hope that our future overlord masters are kind and benevolent, Leo.

Leo: I hope these bots like us. We like you, bots. We really do. That's Steve Gibson. He brings us, see, he brings us, I mean, this show has a payload. This show is loaded like one of them Kinder Eggs with a gooey inside that you just can't get enough of. I'm sorry. That's not a good description. However, I do appreciate everything you do every week, coming up with these stories, talking about it, filling us in on the latest security news.

Steve Gibson, you'll find him at GRC.com. That's where this thing he's working on, SpinRite - I'm going to say, Steve, this is your life's work.

Steve: Yeah.

Leo: You've done many things, but this is your life's work.

Steve: It is.

Leo: He's laboring away on it to create 6.1. You can get 6.0 right now, guarantee yourself a free copy of 6.1 if you go to GRC.com and look for SpinRite, world's finest

hard drive recovery and maintenance utility. Soon to be, I mean, I really like the direction you're taking this, a diagnostic tool, as well, a really useful diagnostic tool.

Steve: Yeah. And you know, Leo, I forgot to mention it, but these slow spots, we're finding slow spots which are really - it really begs the question, why is it slow, and would this mean, for example, imagine a future high-res bitmap SpinRite that showed you the speed response across the surface of your mass storage, and then zeroed in on it and worked on fixing those areas that might be pre-failure.

Leo: Fascinating.

Steve: So there's lots of interesting experimentation coming.

Leo: Lots of ways to go with this. It's a great life project, I have to say. And I think you've got a ways to - I think you're going to be doing this. I see a SpinRite X in your future. I'm just saying, Steve. GRC.com. While you're there you can pick up 16Kb audio versions of this show for the bandwidth-impaired, 64Kb for those with ears. You can get transcripts, for those with eyes who like to read along. Actually that's really useful to get one or both of those so you can read and listen. I think that's a great idea. GRC.com. Plus lots of great freebies. It's a fun site just to browse around.

We have copies of the show at our website. We don't do the 16Kb or the transcripts. We've got 64Kb, yes, and video. That's our unique slice of Security Now!. That's all at TWiT.tv/sn. You can watch us do the show every Tuesday, 1:30 Pacific if we're on time, which we rarely are; 4:30 Eastern; 2030 UTC. You know, anytime you tune into TWiT.tv/live to those audio or video streams, something's going on. Something interesting's going on. Hang out. You know, the show will be around in a bit. TWiT.tv/live.

Subscriptions, of course, available on all the best podcast platforms, a couple of new ones coming soon. But Stitcher, Slacker, Overcast, Pocket Casts, Google Podcasts, Apple Podcasts, you know the drill. Find Security Now!. Subscribe to it. I strongly recommend it. That way you'll have a copy the minute it's available, to listen to at your leisure.

Steve, have a safe week, have a great week, and we'll see you next time on Security Now!.

Steve: Thanks, Leo.