InControl

Description: This week we look at a couple of new zero-days in Chrome and Apple's OSes. We also look at what the U.S. CISA thinks of not only these, but of 15 other problems that our federal agencies seem to be in no big hurry to fix. And we revisit last summer's SeriousSAM vulnerability in Windows which remains under attack. This being the third Tuesday of the month, we'll look back at the second Tuesday to see how that went. Sunday saw a true emergency patch issued by Adobe that probably canceled some Super Bowl plans, and we have an amazingly bad idea for a WordPress add-on.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-858.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-858-lq.mp3

Google has published their 2021 Bounty Report, and their Project Zero has published stats about how things are going there. We have Microsoft removing a popular and highly abused feature of Windows. And then, because nothing else in the past week commanded the podcast's title, I'll wind up by formally introducing GRC's latest freeware which puts its users firmly "InControl."

SHOW TEASE: It's time for Security Now!. This week a new zero-day in Chrome and Apple's OS, and a recommendation from the federal government to fix it fast. Why it was a bad day on Super Bowl Sunday for security experts who are still using an Adobe product. And a new freeware program from Steve that puts you InControl of Windows 10. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 858, recorded Tuesday, February 15th, 2022: InControl. Security Now! listeners, our annual survey is on and going strong. It helps us understand you better, helps us find advertisers who fit your interests. It's optional, of course. But it sure helps us a lot. TWiT.tv/survey22 is the address. Take it now before it closes in March. It'll only take a couple of minutes, and I really appreciate it. TWiT.tv/survey22. Thank you. The following show is brought to you through the generosity of people like you. Thanks.

It's time for Security Now!, the show where we cover your security and privacy online with this guy right here, Steve Gibson from GRC.com. Hello, Steve.

Steve Gibson: Hey, Leo. Great to be with you once again for Episode 858, this middle of February, the short month of the year. A bunch of stuff is happening, but there was no single huge event. So I ended up somewhat self-consciously naming the podcast after my most recent piece of freeware.

Leo: I saw your tweet.

Steve: Called InControl.

Leo: Yeah, your big announcement, yeah.

Steve: So we'll just sort of - I'll announce it formally at the end of the podcast, explain what it is. But we're going to first take a look at a couple of new zero-days in Chrome and in Apple's various products. I mean, again, a common code base, it's good for the developers. It does mean that when you've got a problem, it's everywhere, as is the case here. We also look at what the U.S. CISA, the C-I-S-A, thinks of not only those zero-days, one of which it's saying thou must patch, but of 15 other problems that our federal agencies seem to be in no big hurry to fix. We're going to revisit last summer's SeriousSAM vulnerability in Windows, which has remained serious and under attack. This being the third Tuesday of the month, we'll look back at last week's second Tuesday to see how that went. Also, strangely, Sunday saw a true emergency patch issued by Adobe that probably canceled some Super Bowl plans because, I mean, it was really an emergency.

Leo: Hmm.

Steve: Yeah. And we have an amazingly bad idea for a WordPress add-on. Google has published their 2021 Bounty Report. And, boy, if you're looking for some spare change. And their Project Zero has published stats on how things are going over there. We have Microsoft removing a popular and highly abused feature of Windows. And they never take stuff out. They put a lot in, but it's rare that anything leaves. Anyway, it's going to happen. And then because, as I said, nothing else in the past week commanded the podcast's title, although I think I know what I'm going to talk about next week because I just - I ran across something, it's like, ooh, it's too late to get it in. But there was a very effective deliberate BGP hijack which was leveraged into, like, it made money for the hijackers. And normally we talk about Border Gateway Protocol mistakes. The threat is when they're not a mistake, and one happened. So I think we'll talk about that next week. Anyway, as I said, I'll wind up by formally introducing GRC's latest piece of freeware, which puts its users firmly InControl.

Leo: Good name. And I like it that you made it more than just Windows 11. I mean, you've got other things, too, you can do; right?

Steve: Yeah.

Leo: It's more than just blocking that annoying piece of...

Steve: Well, in fact, I'll explain. My first name was StayPut.

Leo: Which is also a good name. It sounds like it's involved with pets in some form or fashion.

Steve: Yeah. Okay, Windows, now you just stay put.

Leo: You just stay there, yeah.

Steve: But what happened was the concept evolved, and a couple of users said, well, what if I don't want it to stay put, I want it to do this.

Leo: Yeah.

Steve: I thought, yeah, that's a good point.

Leo: Power, it's about power.

Steve: So you want to be InControl.

Leo: Personal agency, exactly right.

Steve: Because as we know it's out of control otherwise.

Leo: Yeah. Windows has a mind of its own.

Steve: Our Picture of the Week is not techie. Often they are. But it was in my pile of images for the podcast, and I just got a kick out of it. For those who are not seeing the video or the notes, we've got two guys, sort of scraggly looking, on an empty plain with the sun in the background and some hills. But, you know, it's like not a lot going on, no cars or anything because it's a long time ago. And one says to the other, "I keep writing 'Stone Age' instead of 'Bronze Age' on all my checks."

Leo: Don't you hate it?

Steve: Yeah, I know.

Leo: Don't you hate it when that happens?

Steve: The age changes, and you just keep thinking you're where you were.

Leo: Of course it's the Information Age. You don't have checks anymore. But we remember those days. It's funny, I don't think I've written a check this year at all.

Steve: I haven't. Although I have, you know, Sue is my bookkeeper person, so she does all that for me.

Leo: Yeah, Lisa probably is printing checks all the time.

Steve: And I think what happened, clearly, was once upon a time we were, I mean, you had to get, like, more checkbooks; right? You'd run out of checks, and you had to get more printed.

Leo: Yeah, I remember those days, yeah.

Steve: So you'd, like, 1995, you were just writing blah blah blah 1995, 1995. Like over and over and over and over so that it just became muscle memory. And then New Year's happens. And you've still got to write some checks. And your hands just automatically write 1995 because, you know, you've done it 300 times.

Leo: Honestly, the problem these days is remember how to hand write. I just, I can't...

Steve: Oh. I gave that up in high school. It's like, what?

Leo: What's my signature again?

Steve: I just started printing because printing is safer.

Leo: I asked my son to autograph - he gave me a knife for Christmas. I said, "Oh, autograph the box so I can auction it when you're famous," which he's getting very close to being. And he signed it. I said, "That's your signature? I can't even - I don't know what that is. It's just m-m-m. Could you just write your name under there so at least somebody will know?" And he said, "Yeah, I know." Because he never learned. His generation, what do they need?

Steve: That's really a good point. I mean, you know, it just isn't a skill that you have to have.

Leo: I said, "You've got to practice your autograph, dude. Get ready."

Steve: It actually is going to get kind of lost, isn't it, over time.

Leo: And then he said, "Nobody does autographs anymore, Dad. They do selfies." And I went, oh, you're right.

Steve: Yeah.

Leo: Never mind.

Steve: Oh, Leo.

Leo: So I took a selfie with him.

Steve: Okay. So we have a high-severity zero-day in Chrome. Just yesterday Google moved Chrome for the desktop up to 98.0.4758.102, to eliminate a high-severity zero-day vulnerability that was spotted being used in attacks. And as usual, Google is closed-mouthed about details, saying only that they're aware of reports that an exploit for, and it was assigned CVE-2022 - wow, a low number - 0609. And not surprisingly it's a use-after-free error, which is what the predominance of errors seem to be these days, in Google's Chrome support for animation. And, yes, exploits exist in the wild. While they were at it, they also fixed seven other security problems, all but one of which were classified as being high-severity. So apparently not zero-days, but yes, the high. And so it goes.

Since exploits for today's vulnerabilities are considered highly valuable by those who have them only until they're known, and since it's a marketplace, right, they may have paid a hefty price from the likes of Zerodium for it, so they're never used in widespread indiscriminate attacks, only against targeted individuals or entities of some kind. So it's unlikely in the extreme that any of us will be a target specifically of any of these problems that get fixed. But keeping our browsers updated is so quick and easy and automatic for most of us. For whatever reason, they are generally rolling these things out, and I seem to always be at the end of the list. But when I looked it said, oh, oh, oh, yeah, we have an update for you. Just restart your browser and you'll be good. And it's like, okay, fine.

So here we are in mid-February, the 15th, with the first of the 2022 Chrome zero-days patched. And as we know, last year Google addressed 16 zero-day vulnerabilities. The first one was on February 4th, and the second one was on March 2nd. So we're kind of like right in the middle of where the first two happened. So this year's starting out a little bit better than last. And we'll see how it goes.

Apple also updated against another zero-day. That was last Thursday, and I did notice a couple of my iDevices said, oh, you're going to put your, you know, log back in again manually. We don't trust your face. So they had to restart. iOS, iPadOS, macOS, and Safari, so across the board, due to a flaw in WebKit that was believed also to be actively exploited in the wild. So far this year Apple is having a bit of a rougher time of it than Google, with Google and Chrome, since this will be Apple's third zero-day that it has patched so far. So, you know, the NSO group sure has been giving Apple a run for their money. It does tell you that our software is incredibly complex. There are problems lurking in software. And when there is enough incentive, and we've going to be talking a lot about patching incentives later on because we're going to be talking about bug bounties and so forth, when there's enough incentive to get into people's devices, so far there's a way.

Much like Chrome, and like so many of the flaws we've been encountering recently, the problem is again another use-after-free vulnerability which grants its attacker arbitrary code execution. And much like Google, Apple is not saying much beyond: "Apple is aware of a report that this issue may have been actively exploited." They credited an anonymous researcher for discovering and reporting the flaw.

So anyway, updates are available for all Apple devices that are receiving them. And my phone wasn't up to speed. I went there, looked under settings, general and then settings and then updates. And it wanted to install, it wanted permission to install 15.3.1, which it

was ready to do. So anyway, if you're worried about your devices, if by some chance you're a high-target person, then, yeah, you'll want to update and close this latest problem.

The CISA, the U.S. Cybersecurity and Infrastructure Security Agency, which, boy, we're going to have to live with that one for a long time, has added that Apple zero-day we were just talking about to its short list catalog of vulnerabilities being exploited in the wild. And for some reason they lit a fire under this one. Or as I was thinking about this, maybe it's just because it's just not a big deal to update it. You could argue that, okay, if there is a vulnerability that is arduous for some reason to patch, well, okay, then you give people a little more time. But this is just say yes to Apple and let it restart your device. So don't wait. And apparently maybe they have some information about where these attacks are being aimed. They may know something that Apple's not saying.

According to the CISA's binding operational directive, and of course everything is acronyms with these guys, so this is BOD 22-01, I guess it's the first binding operational directive of the year, federal agencies are now required to patch their systems against this actively exploited vulnerability impacting, as I said, iOS, iPadOS, and macOS devices. CISA said that all Federal Civilian Executive Branch Agencies - and yes, there's an acronym for that - have until February 25th, 2022. What's the 25th? That's a week from next, or, yeah, a week from this coming Friday, so the 25th. So a week and a half to fully patch against this vulnerability. And again, it must be that they're saying do it now because it's just not difficult to restart your device.

CISA said: "These types of vulnerabilities are a frequent attack vector for malicious cyber actors of all types and pose significant risk to the federal enterprise." And they added that: "Although BOD 22-01 only applies to FCEB" - okay, so that's the Federal Civilian Executive Branch - "agencies, CISA strongly urges all organizations to reduce their exposure to cyberattacks by prioritizing timely remediation of Catalog" - as they call it, we'll talk about that in a minute - "Catalog vulnerabilities as part of their vulnerability management practice."

And what's interesting is that this CISA catalog is a short list. It contains only 16 vulnerabilities for which, like, of all the patches that are out there, they've chosen 16. And that's because they are known to still be actively exploited in the wild. Anyway, as I said, we'll get back to that in a second. But one of them, the "SeriousSAM" vulnerability, as it was called last summer when we first talked about it, last Thursday CISA also asked all Federal Civilian Executive Branch agencies to patch, okay, this is a CVE-2021 because, as I said, from last year, 36934, which is the Microsoft Windows Security Accounts Manager - that is the SAM, Security Accounts Manager - bug that allows privilege escalation and credential theft. And for this one they gave a February 24th patch deadline. In other words, two weeks from last Thursday. Again, get this done.

And in fact it was last Thursday that they made the announcement. So at announcement time their deadline was two weeks from today. And the reason being that this thing's been around since last summer. We talked about SeriousSAM when the news of it broke. As its name suggests, it's serious. It's an elevation-of-privilege vulnerability which was introduced into all Windows client and server editions released since October of 2018, and then up until July of 2021.

Okay. So that meant it started with Windows 10 1809 and the matching server version, which was 2019. And the vulnerability was irresponsibly disclosed via a tweet. So despite not being exploited at the time, as we know, Microsoft designates things that surprise them as zero-days. So they called this a zero-day because it was an unwelcome surprise to them when they read about it on Twitter.

Leo: What, they all have Twitter? What? That's hysterical.

Steve: So, yeah. The problem arose due to an overly permissive access control list, you know, ACLs, which were present on multiple security-critical system files. So that was an oopsie which somehow crept into Windows. You know, like Windows 10, not the first Windows 10, but they'd had a few before that. And that included the Security Accounts Manager database. And you don't want anybody looking at that that's not supposed to be, like anyone other than the kernel. So an attacker who successfully exploits this vulnerability could have full run of the system, able to run arbitrary code with system privileges, which of course allows such an attacker to install programs, view/change/delete data, and create new accounts with any rights they choose, typically all.

Okay. Now, all that said, what might jog your memory, because it did mine, about this one in particular is that simply correcting the overly permissive access control lists does not eliminate it fully. After installing the patch, it's necessary to then manually delete all volume shadow copies of system files, including the SAM database. Since the trouble is with their default permissions, which would be restored along with the files, if those backup files with their permissive permissions could be restored, too.

So we talked about this at the time, which is why I suggested it might jog people's memory that, yes, not only did you need to fix the permissions on the files that are now in use in your system, but Windows is actively maintaining backups as it makes changes, which is how it's able to roll mistakes back and, like, fix things that it tries and don't work. So it's definitely necessary to do that. Otherwise the bad guys could still access the archived files that have the weak permissions. So that has to all be done.

Okay. So all of that happened when we talked about it last July. Believe it or not, here we are eight months later, and the U.S. CISA is finally having to lower the boom on Federal Civilian Executive Branch agencies. Now, and you know, they did that with Log4j; right? Like a couple weeks before Christmas? Trying to cancel Christmas. And that didn't work, either. So I'm not sure what it means for the CISA to say, you know, "Thou shalt update by February 24th," because that may or may not happen. It's a week from next Thursday to get this all resolved.

And you wonder how it can possibly be that a serious vulnerability which received an emergency patch - this was an out-of-cycle emergency that hit on July 20th of last summer. How has that not yet been fixed? Every Windows update since then would have incorporated the fix for it. So can it possibly be that there are U.S. Federal Civilian Executive Branch agencies that have not applied any updates to Windows since last summer? If so, I hesitate to say they deserve what they get, but really at some point.

Okay. So this Top 16 list. I think it's worth taking a look at that list of things that CISA says really must be fixed by those agencies over which it apparently exercises some jurisdiction, even if it's not able to cancel Christmas. As we can see in the table on the next page, and I'll explain what's there for those who can't see, we've got a mixture of old and new vulnerabilities. Actually, sort of a ladder of vulnerabilities through time. And they're recognizable to people who've been following along. Some oldies but goodies, given their CVE dates, which start at 2014. We have one from then, 2014; three from 2015; one from 2016; seven, that was a busy year, from 2017; one from 2018; somehow 2019 sneaked past without any; then one from 2020; and of course the SeriousSAM vulnerability from last summer.

Interestingly, all of our old friends are there. We've got the oldest pair from 2014 and '15 are Apple OS X vulnerabilities. We've got that HTTP.SYS remote code execution from 2015, as well as that D-Link DIR-645 router remote code execution that's been around

forever. And apparently, I mean, the point is these are things that are still being attacked. There are machines still out there expressing some vulnerability to them. 2017 had that SMBv1, you know, the Windows file and printer sharing v1 remote code execution that we talked about, what, four or five years ago now. There's a remote code execution in Office. And then there was that raft of Win32k privilege escalation vulnerabilities. Remember when we had a whole spate of those? That was from that Polar Bear hacker girl. Remember her?

Leo: Oh, yeah.

Steve: Sandbox Escaper. And she had that really cool one-person tent thing, kind of a kayak that you got into and zipped over yourself and hoped that there weren't any actual polar bears around. And, you know, she kept annoying Microsoft with her tweets of just like, oh, yeah, here's another one, here's another one. I'm bored. Here's another one. And they ended up hiring her, so that was good.

There was also, I remember you and I, Leo, talked about this in 2017, a Windows .LNK, a shortcut link file remote code execution. And it was like, what? It's 2017. We're still having problems with shortcuts? Wow. Apache Struts had that improper input validation problem at the end of 2017. And then there was the v3 of the SMB, Server Message Blocks. As I said, when I talked last week about like how many problems there are in these legacy protocols, well, yeah. More of them in 2020.

Anyway, all of our old friends are still there. And what's astonishing is that CISA has them all selected out and listed in this "must fix by a deadline" chart because, as I keep saying, these things are still being exploited. They're the most exploited vulnerabilities today. CISA says, if these were to get patched, things would get much better. And in the far right column is the Patch Deadline. Aside from that one at the top, which is the next Thursday deadline, CISA's giving everybody until August 10th. And when I was looking at that, I was thinking, August 10th, that's a Wednesday. What's that?

The only thing I can think of is that it must have been some integer number of months from the date this commandment chart first dropped upon these federal agencies, probably, what, nine months, I guess? Maybe, who knows when. But in any event, everybody has to have all these fixed by Wednesday, August 10th, which really, you know, patches are available for all of them. So especially Apple OS X, there can't be any machines that have not been rebooted and updated since then. But CISA says that's what people are looking for. So let's hope that happens.

And speaking of patching, Patch Tuesday. Which has really become an industry-wide monthly patching extravaganza. And this month Microsoft did not disappoint. They fixed 51 vulnerabilities occurring in Windows, Office, Teams, Azure Data Explorer, Visual Studio Code, and various Windows Kernel components. Among the 51 defects resolved, 50 of those are rated important, and one is rated moderate. And if you're thinking to yourself, wait, 50 are important, and one is moderate. Were none critical? No. No zero-days. Nothing was critical. And of course you'd be correct to be surprised by that observation. Last week was a rare Patch Tuesday. Microsoft also fixed an additional 19 flaws in their Chromium-based Edge web browser.

None of the security vulnerabilities are known to be actively exploited, though one of the flaws fixed is again what Microsoft calls a zero-day, inasmuch as it was publicly disclosed even though it wasn't yet exploited. So, yeah. Someone just said, hey, this. And Microsoft said, oh, okay, we're going to fix that. It's a privilege escalation bug in the Windows kernel. So it's potentially important. There were also a handful of remote code execution vulnerabilities, so always good to have those gone. And also good that nobody

had discovered them before they were all being patched. One was in Microsoft's DNS server. That earned itself a CVSS of 8.8. SharePoint's server also had an RCE rating of 8.8. Windows Hyper-V had an RCE that scored 5.3. And that HEVC Video Extensions had three of its own remote code execution vulnerabilities with CVSSes of 7.8.

And again, we talked about how this happens in something like a codec. Those are difficult to secure. They are very complicated interpreters which are interpreting compressed tokenized data. And the inherent assumption being made by the people who are writing the decompressors is that what they're being given was written by an authentic compressor. Very hard to catch the flaw in that thinking.

Azure Data Explorer contained a spoofing vulnerability, earning itself a CVSS of 8.1. Outlook contained a security bypass vulnerability, 5.3, as did OneDrive for Android also. Actually it was a little easier to exploit, so that got a 5.9. .NET and Teams both had denial-of-service vulnerabilities, meaning you can crash them, with CVSSes of 7.5.

And the world is not yet done with Microsoft's Print Spooler, which saw four elevation-of-privilege flaws fixed, as well as one in the Win32k driver which got a CVSS of 7.8. And that one, that Win32k elevation of privilege was tagged as being more likely to be exploited because last month a very similar vulnerability in the same place, the Win32k driver, did come under attack shortly after it was patched.

So again, the bad guys, as we've said, they look at what gets fixed. They know there's a window between Microsoft's release of these, and everything we were just talking about with CISA says that in some cases that window's open for a long time. Still, they know that there's an opportunity during which they're able to attack something just patched. So patches are analyzed. They are reverse engineered. The problem fixed is found. Then an exploit is designed, and somebody tries to take advantage of it. So Microsoft is suggesting that this one is likely to be exploited quickly, too. And again, though, these are not affecting most of us. These little high-value vulnerabilities are going to be exploited only in targeted attacks.

I mentioned at the top that Patch Tuesday has evolved over the years to become an industry-wide event. Besides Microsoft, security updates were also released last Tuesday by, in alphabetical order, Adobe - although, again, remember, we're going to get to the Sunday surprise, the Super Bowl Sunday surprise that Adobe dropped after they released all of their patches on Patch Tuesday. Also Android; Cisco; Citrix; Intel; the various Linux distributions from Oracle, Red Hat, and SUSE; Mozilla's Firefox; SAP; Schneider Electric; and Siemens all said, hey, let's put our patches out on the same day.

And you've got to wonder how IT folks get anything done on these second Tuesdays and the days immediately following. I imagine they actually don't. They just probably - and this is why we've sort of established this as an industry-wide patch fest is they're able to say, you know, cancel lunch and not have any away meetings and things, and basically just do nothing but analyze and decide what's safe to do and get that done as quickly as possible because these days we know that the bad guys are going to jump on them immediately and try to take advantage of them.

Leo: Hey, I wanted to ask you one thing. You said, you used the term "release after free." That's a common, you know, after buffer overflows. Right?

Steve: Yeah, use after free.

Leo: Use after free. That's using a section of memory that is no longer in use, basically, that's appointed, that's been released.

Steve: Correct.

Leo: It's kind of like buffer overflow.

Steve: Yeah, so in a system that dynamically manages memory, the idea is that the system figures out when you no longer have any references to something.

Leo: Right. It's garbage collection. It's garbage collection.

Steve: Exactly. And so it releases that. It should invalidate any pointers...

Leo: Right.

Steve: ...to that memory.

Leo: Yeah.

Steve: But in some cases, if code holds onto a pointer, the pointer's no longer valid, but it's still pointing somewhere.

Leo: So that's my question. Because there's different ways of doing garbage collection, of course. Count by reference is very common.

Steve: Yup.

Leo: How would you still have that pointer? It shouldn't be releasing the memory if you still have the pointer. Is that an error in the garbage collection? It must be; right?

Steve: Yes. Yes. Exactly.

Leo: Okay. Okay.

Steve: The so-called "automatic language" is too automatic.

Leo: Yeah.

Steve: So it has worked in a fashion that has created the vulnerability.

Leo: Got it.

Steve: And so when it's getting fixed, that's what they're fixing is that they just said, okay, we don't want to put memory management responsibility on the programmer. Besides, it's dumb; right? We can do a better job. Well, programs have branches, and they've got iterations.

Leo: Right.

Steve: And they've got, you know, and nothing keeps a program from making a copy of a pointer for some purpose that the code doesn't track it correctly. So, I mean, the fact is it's very difficult to do a perfect job of automatically managing memory. It's sort of the holy grail in some sense.

Leo: It's still better than using malloc and remembering to free it and all of that, I would imagine. Maybe not. You're an old-school guy. You really - you use and release your own memory.

Steve: I do. And the thing to remember also is that when a process is running, the OS tags all the allocations as belonging to that process.

Leo: Right.

Steve: So when the process terminates, it's not that there's like memory leakage in a properly functioning system. The operating system will also release all the memory that that process had allocated. In fact, that's one of the advantages of the old-school Unix approach is by, you know, they would launch a process like every time you as a client connected to a server, you spawned a copy of that server for your use. And it really didn't matter if there were memory leaks, you know, in some of the things the server did. But as soon as you were disconnected, that process was terminated, as were all of the allocations that it had made. So it was sort of a self-cleaning system. Microsoft by comparison in their model, for example my IIS code running on Windows server, I've got no memory leaks, despite the fact that I'm doing a whole bunch of stuff all the time. But it's sort of a badge of honor to have a server running for five years and having it consuming not a single...

Leo: No memory, yeah.

Steve: No memory more. Because what you would normally see is over time...

Leo: It would dwindle slowly.

Steve: Exactly, slowly creeping upwards. It's like, what, wait, what did I forget? What am I not releasing?

Leo: Right.

Steve: Because the server is never restarting. It's just running without ever shutting down.

Leo: Right. And you know, C and C++ don't have built-in garbage collection.

Steve: Correct.

Leo: What I did not know is that a lot of C++ tooling includes a third-party garbage collection library. And I bet you that's where the errors are. And then languages like Python, Perl, they all have built-in garbage collection as part of their...

Steve: Yes, and it's those things that are being done for you, I mean, the programmer's not even capable of being aware, right, because it's just like, oh, it's built-in.

Leo: You don't have to care about it, yeah.

Steve: Yeah.

Leo: So you're an advocate for manual allocation and deallocation of memory. I guess you know if you wrote the program. Problem is you get these big teams now. It's easy for you. You wrote the program. You know everything the program does.

Steve: Yes. Yes.

Leo: But you get a thousand people working on an operating system, you can't count on them to know.

Steve: Yes. And also the things I do are not massive.

Leo: Right.

Steve: Like probably the biggest thing I've done is the DNS Spoofability system because it's got all kinds of stuff going on. I mean, it's sending out queries, pseudo DNS queries. It's having to send queries to each of the DNS servers that respond to, I mean, it's massively multitasking.

Leo: Yeah.

Steve: But InControl, that I wrote last week, it's like, yeah, you know, it's 80...

Leo: You know. You know what's going on.

Steve: Yeah, 82K. And it's just like, you know...

Leo: You can actually browse the source code if you're worried and look for a malloc statement. Or you don't have malloc.

Steve: Well, I can use the whole...

Leo: How do you allocate memory in a - yeah, yeah, it's all in your mind.

Steve: It's just the entire thing is in my head at once.

Leo: That's amazing. How do you - so memory allocation in assembler. Does it call a BIOS routine? Or you just actually say I'm going to use this area to this area?

Steve: So that's the other thing is that coding a Windows app in assembler is basically scripting the Windows API.

Leo: Right, right. You call the API, yeah.

Steve: Exactly. And so there's alloc.

Leo: That's right, okay.

Steve: And so I say, you know, I need a K. So alloc a K, and I get back a pointer to the beginning of the allocation. And it's, yes, I have to be careful not to do something outside of the allocation. It's entirely my responsibility not to go further. But there is an arena around the allocation. And so when I attempt to free it, Windows will check to make sure that the end of it is still intact, that there was no overwrite at the end. And, if so, it crashes spectacularly, and then I go figure out what it is that I did wrong.

Leo: At least your crashes are spectacular, and they happen while you're around.

Steve: And in fact if you were ever - if you were a fly on the wall, you would sometimes hear me go "Boom."

Leo: It's like an amateur chemist in his basement. There's never any question if something crashes.

Steve: No. It's like, wow, okay. What hap- now let's go look at this.

Leo: I'm writing to the screen buffer, and it's not looking good, yeah.

Steve: Kaboom.

Leo: Kaboom.

Steve: Anyway, yeah. Okay. So speaking of kaboom, I would imagine that a bunch of people had their Super Bowl Sunday plans ruined, some because the buck stopped with them for the creation and release of an emergency five-alarm CVSS 9.8 patch of a zero-day remote code execution bug in the Magento 2 / Adobe eCommerce platforms, which was being actively exploited in the wild. And on the receiving end of the patch news, the screaming need was to immediately apply this patch that Adobe's engineers had pushed out on Super Bowl Sunday, like within hours all the warnings were, because it was that crucial to ecommerce sites that would have otherwise been compromised before half-time.

So to set the stage a bit, since we've never talked about the Magento ecommerce platform, it's open source and written in PHP. It employs multiple other PHP frameworks such as the Laminas framework and Symfony. The Magento source code is distributed under Open Software License (OSL) v3.0. And after bouncing around and changing hands a few times, Magento was most recently acquired four years ago by Adobe, back in May of 2018, for a cool $1.68 billion. Not a bad price for some volunteer-developed open source software that can be freely downloaded. Wow. Yeah.

And Magento's stats are impressive. As of 2019, that's the most recent numbers I could find, more than 100,000 online stores had been created using the Magento platform. The platform code has been downloaded more than 2.5 million times, unfortunately many of those by miscreants who are scouring the source code looking for ways in.

Leo: I hate those miscreants.

Steve: Those miscreants.

Leo: Miscreants.

Steve: And $155 billion worth of goods have been sold through Magento-based systems during just the year 2019 alone. Magento accounts for approximately one third of all ecommerce happening on the web today.

Sansec, a firm which focuses upon ecommerce security, titled the event on Sunday, when they blogged about it yesterday, they titled it "Magento 2 critical vulnerability CVE-2022-24086." And they explained: "This Sunday, February 13th, Adobe released an emergency patch for a critical vulnerability in Magento 2. It allows unauthenticated remote code execution..."

Leo: Oh, wait a minute. You've been writing Magneto. But I'm looking at this quote.

Steve: Magento. Oh, my.

Leo: It's Magento.

Steve: Anyway, "It allows unauthenticated remote code execution," that is, this flaw, which of course is bad. Or as the Sansec guy said, "the worst possible type." They said: "Actual abuse has already been reported. Adobe" - now, here's the gotcha. I mean, I don't understand this. They wrote: "Adobe has been aware of the issue since at least January 27th" - 2.5 weeks prior - "but decided to issue a patch on Super Bowl Sunday." I said Super Bowl. They didn't. "Which," they said, "is highly unusual. Sansec," they wrote, "expects that mass scanning and exploitation will happen within the next 72 hours." I saw elsewhere people saying within a couple hours.

Under Implications in their posting, they wrote: "This vulnerability has a similar severity as the Magento Shoplift vulnerability from 2015. At that time, nearly all unpatched Magento stores globally were compromised in the days after the exploit's publication." And remember, this is PHP. It's a few lines of PHP. So it's not going to take a high-end reverse engineering rocket scientist hacker to figure out what got patched. And since the cat's already out of the bag, Sansec saw no need not to show the full patch in their announcement, which they did. It's just a reg - I looked at it. It's a regular expression, in a loop, string replacement, presumably to sanitize some dangerous user-controlled input to make it undangerous.

The patching advice itself is sort of interesting. First of all, the trouble affects versions 2.3.7-p2 and earlier, and 2.4.3-p1 and earlier of both the Magento and Adobe ecommerce platforms. They're the same thing essentially. Apparently Adobe paid $1.68 billion to put their name on it. And according to Sansec, who clearly understand what's going on, Sansec said: "Sites running Magento 2.3 or 2.4 should install the custom patch from Adobe ASAP, ideally within the next few hours."

Leo: That's ASAP, all right.

Steve: Yeah.

Leo: Holy cow.

Steve: I mean, during the game, like stop, turn off the TV. You've got, like, no time to get this in. They said: "Sites running a version of Magento 2 between 2.3.3 and 2.3.7 should be able to manually apply the patch, as it only concerns a few lines of PHP source." And they said: "And sites running Magento 2.3.3 or below are not vulnerable." So this was introduced after 2.3, between 2.3.3 and 2.3.later. "However, Sansec still recommends," they wrote, "manually implementing the given patch." You know, and why not? Just do what Adobe wants you to, although you could probably safely - oh. Apparently I said this. You could probably safely watch the big game first.

Leo: On the measure now of how much of a crisis it is. Oh, you can watch the game, go ahead.

Steve: Can I finish this episode? Or do I have to just stop right now? Do I have to hit pause, or can I just wait? So online ecommerce sites and vulnerabilities of course are particularly high-value targets with credit card skimming being the primary goal. That's what they're always doing. These guys are installing skimmers which capture credit card credentials on the fly when people make their purchase.

The most active nefarious group in this space is known as the Magecart, M-A-G-E-C-A-R-T, Magecart group which specializes in targeting unpatched versions of Magento, by any other name, in particular, looking for a way to plant credit card skimmers on the checkout pages of ecommerce websites. The Magecart group, which is actually a consortium of many different card-harvesting subgroups, consistently evolves its skimmers to be more effective and efficient at evasion.

And this has been going on for years. I mean, as long as there's been ecommerce code, they've been at it. For example, in November it added an extra browser process that uses the WebGL JavaScript API to check a user's machine to ensure it's not running on a virtual machine, which helps it to evade researcher detection. And last month, in January, an attack on Segway, actually I saw that, and it almost made the podcast, but there wasn't anything like super standout about it. An attack on Segway, you know, the people who make the rolling things, planted a skimmer by using a favicon that traditional security systems wouldn't inspect. So these guys are sneaky.

Now, Adobe, who is working to downplay the apparent sudden severity of the issue after having sat on it for more than two weeks because that's when the CVSS was, I mean, the CVE was issued, they initially characterized the attacks as "very limited." Okay. Still, 9.8, not good. And issue the update on a Sunday. So on Super Bowl Sunday. But card-skimmer activity is on the rise, and we know how long websites often take to update. And for example, completely separate from this, last week Sansec reported a wave of skimming attacks targeting more than 500 sites, in particular those using outdated and unsupported Magento 1 implementations, which are old. And data from Source Defense found as many as 100,000 ecommerce sites that are still using past-end-of-life Magento v1. So, wow. 100,000 sites still using Magento 1.

Okay. Now, while we're on the subject of things written in PHP, and I'm already on record about how I feel about that...

Leo: By the way, it has garbage collection, I just want to point out.

Steve: Yes, it does.

Leo: Yes, it does.

Steve: You don't need to worry about that.

Leo: We'll take care of it.

Steve: We'll take care of that for you. Yet another WordPress add-on, this one called PHP Everywhere. And Leo, a better name has never been coined. You know? And PHP Everywhere really doesn't sound like a good idea. It was responsible for placing more

than 30,000 additional WordPress sites at risk of remote code execution. And trivial remote code execution, you know, and that's the worst.

Now, the fact that Magento, which we were just talking about, which powers one third of the world's highly targeted ecommerce sites, was also written in PHP, demonstrates that it is definitely possible for sufficiently skilled developers to author secure website code in PHP. I'm not saying it's not. That's not the issue. The issue is that PHP's deliberately and seductively easy-to-use design, which is what makes it so popular, encourages unskilled developers to place their code online. You know, if you had to write it in C#, which has got lots of little pointy sharp bits on it, you know, you probably wouldn't; right? So people don't. They use PHP because, oh, look, I wrote a line, and it worked. Ship it.

In fact, that pithy little slogan I coined a few weeks ago comes to mind: "Most developers stop working the moment their code starts working." We know that there's a big gap between code which works and code which is also secure against attack. And due to the online website environment where PHP typically finds itself, that often leads to trouble. I now have a bunch of stuff at GRC written by other people in PHP. And the only way I was going to ever allow any of that near GRC's network was by putting all of it on its own physical server located behind a physical firewall, I mean, like with wires, that almost completely cuts it off completely from the rest of GRC's network.

I mean, I'm happy to have that stuff. The forums are great. And grc.sc, the little link shortener, there's something called "YOURLS" is a nice little bit of PHP I picked up from someone. But it's all sequestered behind a firewall. So if something gets loosed there, at least I have containment.

Okay. So today we have PHP Everywhere. Get a load of this. I It's like meta bad. The WordPress plugin, this PHP Everywhere, as it's named, has its name because it - I can hardly even say this - it deliberately allows site owners to execute PHP code anywhere on their site. According to the add-on's description: "This plugin enables PHP code everywhere in your WordPress installation. Using this plugin you can use PHP in Pages and Posts." Wait, posts? You can put PHP in posts?

Leo: Oh, lord. That's a terrible - what could possibly go wrong?

Steve: What could possibly go wrong? And also in the Sidebar.

Leo: Oh, yeah, just embed it, what the hell.

Steve: Yeah. Its description says "everywhere." And it goes on to boast. It says: "The plugin also supports different user restrictions and multiple PHP instances. So feel free to just insert PHP in every part of your WordPress site." It's unbelievable. And then he says: "Examples of use: Create custom contact forms and process any kind of data or upload. Generate user optimized content. Customize every little detail of your WordPress installation." And then we add, what could possibly go wrong?

That's right, create a handy-dandy WordPress add-on that encourages site operators who may barely be able to code to liberally litter PHP everywhere around their site. And you, too, can earn not just one, but three of those oh-so-rare CVSS scores of 9.9, as this author did, accompanied by three of your very own CVEs: CVE-2022-24663, 664, and 665, each with a CVSS score of 9.9. As it turned out, PHP Everywhere's functionality, not surprisingly, allowed the execution of PHP code snippets through WordPress shortcodes.

Unfortunately, WordPress allows any authenticated users to execute shortcodes via the parse-media-shortcode AJAX action, and some plugins also allow unauthenticated shortcode execution. As such, it was possible for any logged-in user, even a user with no permissions such as a Subscriber or a Customer, to execute their own arbitrary PHP on a site that had PHP Everywhere - after all, everywhere - by sending a request with the shortcode parameter set to open a block, so [php_everywhere] that invokes that add-on. Then whatever PHP you want, and then you close the block with a [/php_everywhere]. And not surprisingly, executing arbitrary PHP on a site typically allows complete site takeover.

Now, we've been here before; right? Allowing users to execute their own code is reminiscent of all the persistent problems we used to have with SQL injection. That inspired this classic XKCD cartoon. Four frames. We've got the first frame. Mom is listening on the phone. She's picked up the phone, and she hears over the phone, "Hi. This is your son's school. We're having some computer trouble." And Mom says, "Oh, dear. Did he break something?" And the school replies, "In a way," and then asks Mom, "Did you really name your son Robert '); DROP TABLE Students;--?" And Mom says, "Oh, yes. Little Bobby Tables, we call him." And the school says, "Well, we've lost this year's student records. I hope you're happy." And Mom replies, "And I hope you've learned to sanitize your database inputs." Because of course "drop table students" is a command that SQL could execute, and somebody typed it in as the student name, and blammo.

So this is a perfect case in point here. This PHP Everywhere should obviously never have been allowed to happen. But there's no oversight, and the deliberately created ecosystem surrounding WordPress encourages this sort of thing; right? It's like, oh, we create add-ons for WordPress. They're great. And since none of my ranting is likely to change WordPress's approach one iota, I understand, my only hope and my purpose here is to adequately instill in our listeners a sober appreciation for the dangers inherent in using third-party add-ons with WordPress.

It's clear, as you've said, Leo, that the base WordPress system itself is mature, was professionally written, and is being professionally maintained, just like Magento. It's secure and bulletproof. But that security doesn't necessarily pertain in any way to anything that's added to it. And so it's super critical that our listeners keep that in mind, that is, just because the base WordPress is solid doesn't mean some wacko can't create PHP Everywhere and, gee, isn't that convenient. People can put PHP in their posts. Oh, yeah. Log4j, anyone? Wow.

Okay. Google's Vulnerability Reward Program for 2021. Last Thursday, Google shared the results of their vulnerability responsible reporting reward program for the previous year, $8.7 million handed out to people responsibly reporting problems to Google. There are a bunch of interesting facts and stats, and dollar amounts and bug counts, that I believe our listeners will be interested in. So I'm going to share an edited-down version of what Google wrote.

They said: "Last year was another record setter for our Vulnerability Reward Program." Those are VRPs. They said: "Throughout 2021, we partnered with the security researcher community to identify and fix thousands of vulnerabilities, helping keep our users and the Internet safe. Thanks to these incredible researchers, Vulnerability Reward Programs across Google continued to grow, and we're excited to report that in 2021 we awarded a record-breaking $8,700,000 in vulnerability rewards, with researchers donating over $300,000 of their rewards to charities of their choice.

"We also launched bughunters.google.com," and that's a URL I wanted to be sure to share with our listeners who might be interested last year, "bughunters.google.com, a public researcher portal dedicated to keeping Google products and the Internet safe and secure. This new platform brings all of our VRPs - Google, Android, Abuse, Chrome, and

Google Play - closer together and provides a single intake form, making security bug submission easier than ever. We're excited about everything the new Bug Hunters portal has to offer."

Then they talk about some specifics. For Android, "The Android VRP doubled its 2020 total payouts last year [2021] with nearly $3 million in rewards, and awarded the highest payout in Android VRP history, an exploit chain discovered in Android receiving a reward of $157,000. Our industry-leading prize of $1.5 million for a compromise of our Titan-M Security chip used in our Pixel device remains unclaimed." Of course that's good news. They don't want to pay that out. But they want to say, hey, if you find something wrong, how about 1.5 million bucks? They said: "For more information on this reward and Android exploit chain rewards, please visit our public rules page."

They said: "The program also launched the Android Chipset Security Reward Program, a vulnerability reward program offered by Google in collaboration with manufacturers of certain popular Android chipsets. This private, invite-only program provides reward and recognition for contributions of security researchers who invest their time and effort into helping make Android devices more secure. In 2021 the ACSRP paid out $296,000 for over 220 valid and unique security reports." So not high payout individually, but probably easier to get. So some lower hanging fruit.

They said: "We'd also like to give a special shout-out to some of our top researchers whose continued hard work keeps Android safe and secure: Aman Pandey of the Bugsmirror Team," they said, "has skyrocketed to our top researcher last year, submitting 232 vulnerabilities in 2021. Since submitting their first report in 2019, Aman has reported over 280 valid vulnerabilities to the Android VRP" - and again, just 232 last year, so he's really accelerated his pace - "and has been a crucial part of making our program so successful." He probably made himself some nice money, too.

"Yu-Cheng Lin has been another phenomenal researcher for the Android VRP, submitting a whopping 128 valid reports to the program last year. Researcher" - and we just have a wacky Gmail email - "discovered a critical exploit chain in Android, receiving the highest payout in Android VRP history." Oh, he's the guy who got the $157,000 for an exploit chain.

For Chrome they said: "This year the Chrome VRP also set some new records. 115 VRP researchers were rewarded for 333 unique Chrome security bug reports submitted in 2021, totaling $3.3 million in VRP rewards. The contributions did not only help us to improve Chrome, but also the web at large by bolstering the security of all browsers based on Chromium." And they call out some other leading researchers, and that's pretty much it. "So a huge thanks and congratulations to all Chrome VRP researchers that helped us make Chrome and Chrome OS more safe."

So $8.7 million last year. You know, bug hunting, we've talked about it a lot, is not guaranteed income, but it might be an interesting way to spend some spare evenings when nothing else is going on. The more you look and poke around, the more you'll learn. And as I've said before, and every time it's happened for me it's been the case, there is really no better way to extend one's own coding skills than by reading and comprehending someone else's code. You just, you look at it, and it's just, you know, it's almost like somebody else wrote it. You learn things. And you might have a well-earned payday.

Leo: You also learn a lot about coding by looking at other people's code. It's valuable. Everybody's got a different way of doing things.

Steve: Right, well, and when you think about it, how do writers learn to write? They learn by reading.

Leo: Exactly. They read. Yeah.

Steve: Yeah.

Leo: It's challenging, at first. Everybody has a different way of coding.

Steve: Yeah. Oh, and finally, before we wrap, we have Google's Project Zero stats which I thought were also interesting. Okay. So this is of course their zero-day flaws which were found across the industry. Their recently published data, the good news is, reveals that the average period software vendors used to repair and issue security updates reported by Project Zero last year was 52 days, which was a significant reduction despite no reduction in bug levels from 80 days three years ago. So three years ago 80 days average to fix things. Now we're down to 52. And again, not like there were fewer problems. And now nearly all vendors are addressing their flaws within what has become an accepted industry deadline of 90 days. And then after that, just before lowering the boom, for some reason there's an agreed-upon grace period of two weeks. I don't quite understand that, but okay.

The stats for the 2019-2021 period show a total of 376 zero-day reports from Project Zero with 26% concerning Microsoft, 23% Apple, and 16% Google. The sum of those three is 65%. So those three Microsoft, Apple and Google constitute essentially two thirds of all of the Project Zero reports. And it's not surprising that the largest commercial desktop OS and the vendors of the two biggest mobile OS vendors or the providers of the two biggest mobile OSes would be those top three.

For anyone who's interested in a complete breakdown, I've got a chart in the show notes which breaks the stats down by vendor. Oracle had the fewest Project Zero reported zero-days, I mean, and really remarkably few at only seven, seven in 2021; although they also took the longest to fix those seven, with four of them exceeding both the 90-day fix-by deadline and the additional two-week grace period. So, you know, I guess that's one advantage of having lots of problems is you end up building a team that's, like, awake all the time. Whereas Oracle's people probably, like, have to be brought back in from vacation in order to fix the problem.

And as the chart above shows, the three overall best patching performers by average days to fix, and impressively, were Linux at just 25 days, they're the minimum average days to fix a problem; Google at 44 days; and Mozilla at 46. At the other end, the worst performing was Oracle, as I said, at 109 days. But again, they only had seven problems compared to Apple, who had 84 zero-days. So then we have Microsoft, who took an average of 83 days each; and Samsung, who was a bit quicker at 73. And notably Microsoft also had the most fixes, 15 of them, which occurred within that final two-week grace period. So Microsoft was blowing through their 90 days. And just before Google was saying, uh, you know, we're going to tell everybody what's wrong here...

Leo: That's how I do my homework. I wait till the very end and then, yeah.

Steve: Exactly. And comparing the mobile OS terrain, iOS and Android are about tied, with iOS having an average time-to-fix of 70 days and Android taking 72. And on the web

browser side, not surprisingly, since we often observe Google's quick response with Chrome, they beat everyone with an average time-to-fix of 29.9 days average, so like 30. And on the other end is Apple's WebKit, the longest at 72.7 days on average. At the same time, that was 40 bugs for Chrome and only 27 for WebKit.

Leo: Look at Firefox, only eight.

Steve: Yeah.

Leo: Yeah.

Steve: Yeah, Firefox did a really nice job last year. And again, as an industry overall, it seems clear that, despite the appearance of an increasing rate of problems and it feels like an increase in problem severity since we're dragging an ever-growing legacy behind us still those tasked with fixing problems, much as we might wish they were even better, they're managing to stay ahead, keeping things under control and reducing our overall exposure.

And you know, the one thing I was tempted to do was to multiply the number of days of an outstanding problem by the number of problems because that would kind of give you the, I don't know what you'd call it, not really the area under the curve. But that's also sort of an interesting metric. And like Apple, who took twice as long, had half as many problems. So the problem exposure area was sort of the same.

So anyway, one last thing. Bye-bye, WMIC. And yes, K-E-Y-M-O-U-S-E. Last week we talked about the practice of "Living Off the Land," where malware or, Leo, miscreants take advantage of commands and features available at the local system, where they find themselves. One of the more often abused features of Windows, which we touched on last week, is the massively capable Windows Management Instrumentation (WMI) system, which is accessible through the command-line executable wmic.exe.

The fact that this useful command-line access tool is so often abused has not escaped Microsoft's attention. As a result, wmic.exe is going away. And if you know anything about Microsoft, it's that they really, really, really, really, really, really strongly really dislike ever removing anything from Windows, especially something like WMIC, a system management tool that almost certainly figures actively into the management scripts being used by their enterprise users, who they care about most, among others.

And despite this extreme reluctance, Microsoft is in the process of removing wmic.exe, starting with the latest Windows 11 preview builds in the Dev channel. They did announce last year that this was their intention, and that they were in the process of deprecating the use of wmic.exe in Windows Server in favor of Windows PowerShell, which provides a full superset of WMIC's capabilities, including the ability to query the Windows Management Instrumentation system directly.

Microsoft wrote: "The WMIC tool is deprecated in Windows 10, version 21H1, and the 21H1 General Availability Channel release of Windows Server. This tool is superseded" - okay. But whenever Microsoft supersedes something, they leave the other one alone; right? Still there. They said: "This tool is superseded by Windows PowerShell for WMI. This deprecation only applies to the command-line management tool. WMI itself is not affected."

Okay. So now Microsoft has been spotted removing WMIC from Windows clients, starting with Windows 11 preview builds in the Dev channel. The guys at BleepingComputer did some sleuthing and confirmed that, from at least build 22523, the WMIC command is no longer available in Dev channel clients, although they noted that Microsoft may have removed it from earlier builds which they didn't check.

Microsoft may be testing the waters to see whether they're able to pull it from Windows, or to gauge how much fur will fly if they do. They clearly want to, and you'll note that the only possible reason for deliberately yanking something from Windows that's already there and that many good people are using, is that many bad people are using it, too.

One thing that might happen, which wouldn't surprise me, would be for them to make it available separately as an optional manual download. That's something they've done before. It's certainly the case that 99.999, and you can just keep on going with nines, percent of Windows client users have never typed the command "wmic," and never will. So having it there, only to ever be used by malware, really makes no sense. But leaving it as optionally available, though deprecated and not recommended, minimizes the damage from pulling it from existence entirely. So it won't be present, but power users will be able to get it. Or enterpriser users could selectively get it.

And you know, Leo, this makes me question Windows' future. It is so loaded with that kind of thing that nobody who looks at the friendly candy-coated surface of Windows ever knows about. But it's there, and it's being abused. You kind of wonder whether there might at some point be like a power tools package that you could easily install, but where just generic Windows won't have it because only the bad guys are ever going to use it, never the person sitting in front of the keyboard. In any event, for maximum compatibility, everyone should take this as a heads-up. If you or your enterprise are currently dependent in any way upon wmic.exe, you'd be well advised to move over to PowerShell.

Back in 2016, everybody was upset over Microsoft pushing us to Windows 10, so I created Never10. Pithy little name. I liked it a lot. Three million people have downloaded it.

Leo: Are you kidding?

Steve: Three million.

Leo: Holy cow. Microsoft must hate you.

Steve: Yeah. They're probably thinking, what's he going to do now?

Leo: Oh, my god.

Steve: Yeah. And really they're upsetting people, right, by now what they're doing is upgrading your Windows without your knowledge, people complaining about it constantly.

Leo: Yeah.

Steve: And you look at Windows 11. Well, you can't put the task bar on the side of the screen. I've looked at it, Leo. I had to have a laptop running 11 in order to get this latest piece of freeware done. I mean, it is beautiful looking. I have to say, I mean, it's gorgeous. But I'm happy with Windows 10. And yes, somebody's somewhere saying "When pigs fly, Steve is happy with Windows 10." But it's true.

Anyway, so first I thought, okay, it really isn't Never11 because people are also not wanting their versions or their feature releases of 10 to change. So people who are on 10 want, basically, people want to stay put. And so that was like, oh, I'll call it StayPut.

So then as I started talking in the newsgroup with the gang who helped me over the past week to get this thing all polished and honed and working just the way we wanted, someone said, well, that's good, but it won't really work for me because I do want to move Windows 10 from 21H1 to 21H2. So I'd like to do that.

Well, as I looked deeper into Microsoft support for group policies, I realized that the way they had implemented the controls, targeted at enterprise users, but available to everyone, was literally with a targeted release version in the registry. So that allows a user of Windows to target Windows Update to a specific major version, like 10 or 11 or 12, and then the individual feature release within that major version, which of course is now what Microsoft is doing. And if you target at the one you're on now, well, then, you get the advantage, you get the equivalent of staying put. It will not go.

And in fact during my testing I had a machine - how did I run across this? I can't remember now. I had a machine with like 1504 or something, like from the dawn of Windows 10. And I put InControl on it, and boy, was Windows Update unhappy. I mean, it was like squirming around because it had gone out of support a long time ago. But it would not update it. So this lock is powerful. But the advantage is it doesn't shut down Windows Update. All your security updates, your monthly standard roll forward continue. But it will keep you at where you are at the major version and the feature release, either where you are, if you just leave it set to its default, or if you wanted to say, yeah, I'm ready to go to Windows 11, you're literally, you can click the button to release control.

There's two fields in the lower left for Version and Release. You could change the 10 to 11 and hit Take Control, and it would lock it, telling it wants Windows Update to take you to Windows 11 as soon as it will. So anyway, it went from StayPut to UnderControl to TakeControl to InControl, which is where we are. And as I said before, 82K of assembly language, just one of my little cute, you know, you don't have to set it up or install it or anything. You just run it. And the only thing I have left to finish is the documentation. As I did for Never10, I want to fully document the six Registry keys. There are six Registry keys that it manages. And that's all it is. It's simple. But there are, as I said last week, there are sites that are telling people to disable Windows Update completely,

Leo: No. No. Yeah, no.

Steve: I know. It's like bad, bad, bad, bad, bad.

Leo: Unh-unh, yeah.

Steve: So this gives people some control that they wouldn't otherwise have in a very friendly user interface.

Leo: Good.

Steve: So that's it. And that's our podcast.

Leo: And that's the show, ladies and gentlemen. Every week he comes up with something. Not always a program, but that's cool. It just changes the Registry. And I presume if it doesn't see that key, it doesn't screw with it or anything like that. Doesn't create...

Steve: Correct. And in fact, some people have a few of the things already set.

Leo: Right.

Steve: And so the other thing, it'll come up and say, well, you are partially in control.

Leo: Oh, that's...

Steve: And it will say you're missing three of the six Registry keys required to blah blah blah.

Leo: That's great.

Steve: So, you know, it does the whole thing.

Leo: Very nicely done. Steve is at GRC.com. There's lots of reasons to go there besides InControl, although that's a great reason. There's also of course ShieldsUP!, his test for your router to make sure that you are properly set, even more important these days than ever before. And his bread and butter, SpinRite, the world's best mass storage maintenance and recovery utility. Currently version 6 is out there. If you buy it, you'll get an automatic upgrade to the next version. You'll also get to participate in the development of that version. GRC.com. You can leave feedback for him there at GRC.com/feedback. But you can also go to Twitter and leave him some feedback. He is @SGgrc. And DMs are open now, ready for your call.

While you're at the website, you might want to check out his version of the podcast. He has two unique versions, ones that we don't have, a 16Kb version which is a little scratchy, but it is the smallest audio version available. And so if you've got limits on your bandwidth or you're close to your bandwidth caps, that's a good choice. There's also, even smaller and really useful, the transcripts, which are done by a human. Our transcripts are AI. He does it, he spends some money and gets somebody real to do it. And so they're very, very good, and I can't wait to see the transcripts from this week. Thanks to Elaine Farris for doing those, and thanks to Steve for paying for it. That's a really nice service. All at GRC.com.

We have audio and video for the show at our website, which is TWiT.tv/sn. SN's for Security Now!, obviously. You can get it there. You can also go to YouTube. There's a channel dedicated to Security Now! with all the videos there. That's an easy way to

share it with somebody else. And of course you can subscribe in your favorite podcast player. Get it the minute it's available. And if you would, leave us a review. Let the world know about Security Now!. I think that everybody needs to know about Security Now!.

Steve, we'll be back here Tuesday, about 1:30 Pacific, 4:30 Eastern, 21:30 UTC, for another gripping edition of Security Now!.

Steve: For the final episode of February.

Leo: Yes.

Steve: As we've got a short month this month.

Leo: As we mosey on through to 999.

Steve: Yeah, my birthday month.

Leo: Hey, thank you, sir. Have a great week. We'll see you next time on Security Now!.

Steve: Thanks, buddy.