SECURITY NOW!

Transcript of Episode #814

PwnIt and OwnIt

Description: This week we start with some needed revisiting of previous major topics. We look at an additional remote port that Chrome will soon be blocking, and the need to change server ports if you're using it. We look again at Google's forthcoming FLoC non-tracking technology and a new test page put up by the EFF. We revisit the PHP GIT server hack now that it's been fully understood. We look at Cisco's eyebrow-raising decision not to update some end-of-life routers having newly revealed critical vulnerabilities, and we also examine another instance of the industry's failure to patch for years. Then, we conclude with a blow-by-blow, or hack-by-hack, walkthrough of last week's quite revealing and somewhat chilling Pwn2Own competition.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-814.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-814-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with the answer to some critical questions. Why are Firefox and Chrome blocking port 10080, hmm? We'll also talk about FLoC, Google's Federated Learning of Cohorts technology. It's starting to roll out now, and he has a good way of figuring out if you're in the FLoC. And then it's a look at Pwn2Own, some fun exploits, some not-so-fun problems. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 814, recorded Tuesday, April 13th, 2021: PwnIt and OwnIt.

It's time for Security Now!, the show where we get together with this guy right here, Steve Gibson, every week and talk about your security and privacy online. Okay. I'm looking at the salute, the Vulcan salute. There's something different. I'm trying to figure out what it is about that ring finger of yours. What's going on, Steve?

Steve Gibson: Looks a little bit like yours, actually, Leo.

Leo: It does.

Steve: Yeah.

Leo: There's a band of gold on it.

Steve: Yeah. After three and a half years of Lorrie being very patient, I decided that she had been patient enough. Actually, about a month ago we decided that we were going to make it official. And she's so much more than a girlfriend. At one point she said, "So, what am I, your girlfriend?" And I thought, that's not right. So...

Leo: Yeah. At our age, "girlfriend" doesn't really say it, does it.

Steve: Yeah, yeah. And if you say, oh, my "partner," then they're like, well, [crosstalk] this partnership and so forth.

Leo: Are you gay? Yeah, exactly, yeah. "Wife" has a certain - everybody understands what a wife is. It's a certain ring to it.

Steve: It's a done deal, yeah. It is a four-letter word, but it's also a good one.

Leo: Good, good, good. Congratulations.

Steve: So anyway, we made it official last Wednesday afternoon, just a very quiet little ceremony. We actually ordained her son with the Universal Life Church.

Leo: Oh, that's perfect.

Steve: Where you click "Ordain Me" on the website.

Leo: Yeah, I have that, I have that, yeah. And you're legal.

Steve: Yeah. And she wrote our vows...

Leo: Oh, nice.

Steve: ...which her son read, and we did our best to repeat.

Leo: Oh, that's great.

Steve: And it was really - and just kept it quiet to ourselves. And so it feels really good. And here I am, what, five days in.

Leo: Congratulations, yeah.

Steve: And not a regret, no doubt or anything.

Leo: Married man. That's very nice. That's really great, yeah. Congratulations. She's great, too, by the way. We've met, and I gave her my seal of approval some time ago.

Steve: Yeah, she is. Well, and I told a story, I posted the news over on the newsgroups, and I said that at this time of your life, by this point, you've probably pretty much figured out who you are. And so on our first date she asked me one of her, like, test questions. And she said, "What are your plans for retirement?" And I thought, you know, let's just, you know, no reason to soft pedal this. Let's just rip the bandage right off. If this is going to be the first and last date, then better to know now. So I said, "Oh, I'm never going to retire." Which turned out to be the right answer.

Leo: The right answer, good. It's a good answer for me, too, by the way, Steve. I just want to - glad to hear it.

Steve: Yeah. And I explained that every relationship the friction had been that I just love what I'm doing so much. And that had always been a problem. And so I think basically I waited for three and a half years because I couldn't believe it, that it was really possible to have someone who let me work.

Leo: That's wonderful. That's great.

Steve: And yeah, so, anyway. Very happy.

Leo: Congratulations. And she's great, and you two make a wonderful couple. So I'm very happy for you both.

Steve: Thank you. So PwnIt and OwnIt for Episode 814, here in mid-April. This is when the annual Pwn2Own competition occurs. And ever since they began we've had a lot of fun. In years past you sort of gave us the, what was his name, Howard Cosell, the Howard Cosell read.

Leo: Howard Cosell play-by-play of Pwn2Own. It's an amazing battle.

Steve: And I made it the topic because it's a little bracing just to watch these guys cut through what we think of as and hope are secure systems, just like, eh, figured I'd make an extra 40 grand, so here's your escalation of vulnerability, or your escalation of privilege. Or, I mean, and oh, Leo, you would think by now that maybe like Exchange Server was safe? Unh-unh. No, no, no.

Leo: What? There's more?

Steve: Oh.

Leo: Oh, my god.

Steve: So we're going to start the week with some topics that we need to revisit because we've been picking good things recently where they keep generating more interesting news. So we're going to look at an additional remote port that Chrome will soon be blocking, and perhaps the need to change some server ports if any of our listeners happen to be using the one that Chrome has decided it now needs to block.

We're going to look again at Google's forthcoming FLoC, their non-tracking technology, and at the new test page that has been put up by the EFF, who doesn't like it. Of course, they don't like anything. Then we're going to revisit the PHP GIT server hack, now that we understand fully how it happened. And we're going to look at Cisco's eyebrow-raising decision not to update some end-of-life routers that have newly revealed critical as in remote access, vulnerabilities. But they've said they're old, so no.

And we're also going to examine another instance of the industry's failure to patch for years, and the consequences. And then I think a fun blow-by-blow or hack-by-hack walkthrough of last week's quite revealing and a little bit chilling Pwn2Own competition.

Leo: And I promise not to do my Howard Cosell for it. All right. Good show.

Steve: And we do have - this Picture of the Week I've been sort of sitting on for a while. But I just, for our audience, in context, it's like, really? Like, no.

Leo: I've been laughing at that one for a while. To the Picture of the Week, Mr. Steverino.

Steve: So when I was putting this - I grabbed it from my stock of pictures for the podcast. Sometimes I run out; sometimes I have a little excess. I've been waiting because we've had more relevant pictures recently. And I put this on the show notes, and Lorrie walked by. And I said, "Look at this. This is like - what does this say?" And she said, "That's not real." I said, "Oh, yeah. They actually said that." Anyway, this is the big blue, I don't know what color it is, aqua-colored screen that we all see now who use Windows 10.

Leo: It's Windows Blue. We all know.

Steve: Windows Blue. That's right. And anyway, it just - this is purely rhetorical, of course. But this is a screen that comes up, I don't remember when, like maybe when it's doing a feature update or something.

Leo: Yeah, so annoying. This goes along with "All your files are right where you left them."

Steve: Yeah.

Leo: It's just - but it tells you a little. It is actually a little bit of - I think it tells you a little bit about Microsoft's mindset, frankly; right? Leave everything to us.

Steve: Leave everything to us. Just stand back from your computer.

Leo: Just let us do it.

Steve: And down below it says, "Don't turn off your PC."

Leo: Oh, yeah, don't do that.

Steve: We're busy. So, but leave everything to us. And of course which begs the question that we often ask on this podcast: What could possibly go wrong?

Leo: I love it.

Steve: Anyway. And, you know, Lorrie's been abused by Windows 10. It does the same things to her that it does to everybody else. She's like, "Really? They're actually saying that?" Uh-huh, yeah.

So as we know, the practice known as NAT Slipstreaming, which we've talked about twice recently, uses or rather abuses the user's local router's Application Layer Gateway features. Through NAT Slipstreaming, code running on a browser inside the LAN is able to arrange for unsolicited traffic from outside to get into the LAN and to go to specific devices, thus effectively bypassing the natural firewall features that we love so much and depend upon our NAT routers for. For this form of abuse to work, the browser needs to be able to emit traffic bound for specific remote ports.

So browsers have been fighting back, thus our two previous mentions of this, by preventing outbound connections to specific abuse-prone ports which NAT routers may be monitoring for application layer traffic. Of course famously FTP was a big one, where because of the trickiness of the FTP protocol, active FTP, a packet outbound would itself in the packet carry information for the remote server, telling it which port to return its traffic to.

So the NAT router, for NAT to work with active FTP, it had to be inspecting the packet's contents, realize that this is FTP, look inside it to see if it's got that port that the remote server is being told to connect back on because then it needs to proactively open that so that when the remote server does respond on that port, it'll be able to come through the NAT and get back to the proper computer inside the LAN. So there's a sort of a simple example. But many of the more fancy, more modern, if you will, protocols have things in their actual protocol that affect the traffic. So NAT routers need to be able to listen to that.

At the moment, Chrome is blocking ports 69, 137, 161, 554, 1719 and 1720, 1723, 5060, 5061, and 6566. We're talking about this because last Thursday Google stated that they intend to add TCP port 10080, so 10080, to their growing list. And as it turns out, it's been on Firefox's block list since late last year, November of 2020, when it got added for Firefox. That 10080 port is known to be used by the Amanda backup software and also by VMware's vCenter. But in neither of those two cases is there any need for a web browser to be initiating traffic to it, to those ports. So no reason not to block it at the browser. Well, almost none.

The biggest concern is that very much like port 8080, which is a popular port of choice for so-called userland web browsers, you know, back in the dawn of the podcast we talked about how in the original Unix implementation of IP networking, ports 1 or 0, depending upon how far down - there technically is a zero because you could have all the bits, all 10 bits of the port number could be zero. Oh, wait, all 16 bits, sorry. I was thinking of 10 bits because that's where the kernel access ends.

Ports 1 through 1023 are reserved so that only processes running with root privilege are able to open ports, those low-numbered ports. As a consequence, it's not possible for a user to run their own web browser on port 80. Being a user, they don't have access to anything below 1024. So but port 8080 is often used for that. Well, so is 10080, just because it has, you know, it's a nice high number that ends in 80, which of course is historically HTTP.

And when I was researching this issue last night, I encountered some instances of problems. There was a posting on Reddit that was titled, "Why Firefox Developer Edition blocks HTTP on port 10080 after update." And this guy wrote: "My current school project uses port 10080 to run the web application. After the update to Firefox 85b2 Dev channel," he said, "I need to set network.security.ports.banned.override in order to access my web. What's the reason behind the ban of this port? I'm curious because it's not a standard port. Firefox allowed it before, and it's not used by any widely known applications." He's right on all counts.

So of course we've talked before about how dicey it can be to ever take back anything once it's been given. The Internet, the web, and many of its descendent technologies were originally designed by technologists who placed very few and in retrospect, too few limitations on the use of the new toys that they were creating. And look at how cautious, for example, Google was with Chrome's careful gradual tiptoeing removal of FTP. I mean, they very much wanted to remove it and just not have an FTP client in their browser. But they were also very worried about taking anything away that users might be using and depending upon.

So over in the Bugzilla forum, Mozilla's forum, someone posted: "I have a similar problem with this change. I have thousands of CPE" - and I'm sure he was referring to Customer Premise Equipment with the standard abbreviation - "with their management port on 10080." He says: "Of course not accessible from the Internet. But I need to manage them. Is there any option in about:config that will allow me to use port 10080 again?" And of course he was referring to this change which hit him also in Firefox last November. And somebody at Mozilla replied, saying: "Sorry for the inconvenience this is causing." And they had a three bullet point instruction: "Go to about:config. Create a new pref of type String with the name network.security.ports.banned.override."

And first of all, I didn't even know you could do that. I thought, like, all of the preferences were built in, and you search for a piece of one in the search bar, and then you get them. But when I went and looked, it's like, oh, sure enough, there's a plus sign over on the right. You can create your own. So cool. Anyway, and the third instruction is add the required ports as the preference value. He says: "You can also add multiple as a comma-separated list or as ranges." And he says: "The change does not require a restart."

Leo: We should point out you can create your own, but it might not do anything.

Steve: Exactly. If you say, you know, wave.hi.to.mom, then...

Leo: It's not going to do anything. Clearly Mozilla's code does check to see if such a preference exists.

Steve: Yeah, you could, like, maybe discover a cookie that nobody knew was in there; right?

Leo: Yeah, I bet there's a lot of stuff in there, yeah. Yeah, who knows?

Steve: Don't spend your time.

Leo: No.

Steve: So as far as Chrome's decision goes, their developer, Adam Rice, noted of port 10080, he said: "It is an attractive port for HTTP because it ends in '80' and does not require root privileges to bind on Unix systems." To allow developers to continue using the port, he said that they'll be adding an enterprise policy that developers can use to override the block. "Once Chrome's change is in place" - that will happen at some point. I didn't see any version number, but imminently - "users will receive an 'ERR_UNSAFE_PORT' message whenever Chrome attempts to access remote port 10080." So the takeaway for our listeners is that it would probably be a good idea to migrate any services you might be hosting on port 10080, which need to be accessed from web browsers, move them to, you know, what, 9080 or something.

Leo: Or 10081, you know, I'm sure there's plenty of places. And there's 65,000 places you can go.

Steve: Exactly.

Leo: But my question is, and I don't think anybody's answered this, why 10080? Is there something going on in 10080 that they're trying to prevent?

Steve: The reason I ran across those other references was I was asking the same question of the great Internet, and I was unable to find an answer. So it is not a widely used port. The only reason, you know, it fell under the category of NAT Slipstreaming, so it must be that there somewhere is a router that matters, that Google became aware of, that is abusable if it sees traffic passing by it to 10080. Not any of our consumer routers because, as we said, those are generally low-numbered ports that are bound to services that need some special treatment. So perfect question, Leo.

Leo: I think I found it. You're the one who should know. You've got ShieldsUP!. I'm sure you test that port; right?

Steve: Right.

Leo: It actually shows up as a really good resource for looking at what different ports, especially the canonical ports, are being used for. Apparently 10080 is used by Amanda.

Steve: Yes, there is an Amanda backup software that uses it.

Leo: Which is broken now. Sorry, Amanda.

Steve: No, because it's not...

Leo: Is it inbound or...

Steve: That's a very good question. And when I was digging around, I came away thinking that browsers didn't need to access that. But maybe that is Amanda's web interface, and Google just said, well, sorry, as you said, sorry, Amanda, but this is more important. So and it must be because even Adam recognizes that there may be some breakage.

Leo: There is a command injection privilege escalation exploit, CVE-132794, for Amanda. So I wonder if this was to mitigate that.

Steve: Interesting. Maybe they are preventing that abuse of that through Chrome.

Leo: You could use Shodan; right? You could scan for that port on Shodan and see what happens.

Steve: Oh, yeah. And you would find everybody who had it open. And we ought to also note that for some reason Firefox added that. Oh, in fact, it does say Amanda on the Firefox list.

Leo: Yeah. This is an old - it's a 2019 CVE. I'm sorry, 2016 CVE. A user with backup privileges could trivially compromise a client installation, and that might be why. But you would think they would want Amanda to fix the exploit as opposed to blocking the port for everybody.

Steve: Yeah, exactly, like block it for everybody. And Leo, in the show notes I do have that searchfox.org/mozilla-central. If you click that, because I saw that there is a #98, that'll jump you right to the top of Mozilla's huge list of blocked ports. It's like way more even than Chrome is blocking.

Leo: Wow. Interesting.

Steve: So, yeah, they're really shutting down a bunch of ports on the outbound side. And you'll notice at the bottom there...

Leo: There's Amanda.

Steve: There it is, and it's labeled Amanda.

Leo: Yeah, huh. Interesting. So all of these other ports are used for exploits, too; right? Why else would you block them?

Steve: Yes. Well, but the other thing is that, as we know, when a client, any client of any of our operating systems says give me an outbound port, they start being allocated on local ports from 1024 and up. So all of those services are going to be down low. And it would, of course, as we know, web browsers by default aim their traffic at 443 for HTTPS. If you can force them over HTTP anymore, that'll be port 80. But you can put colon and then the port number in some script, JavaScript or on your page, to go pull a resource from any port you want to remotely. So anyway, it does look like they've locked down one more problem.

Leo: And Mozilla points to spec.whatwg.org for a port-blocking spec that has all these ports on it. So there's some Internet group - it does not have, I noticed, 10080 on it yet. But there's some group on the Internet that's saying these ports should be blocked due to a bad port. And I don't know what that all means.

Steve: Wow.

Leo: So these are the bad ports. Don't use those. Bad port.

Steve: So believe it or not, this abbreviation of Google's is just - it's not going down easily. I mean, we figured out, we kind of reverse engineered it. Well, you jumped on it because you realized that they had all these bird-themed things over at Google Projects. And so of course they had to use FLoC for their tracking or their anti-tracking abbreviation. And so then if you have FLoC of birds, then you're going to have to reverse engineer, which is why we get the really awful Federated Learning of Cohorts, which, wow, you know.

So we ask rhetorically, are you FLoCed? Or rather, actually, the EFF actually did ask that. Our podcast three weeks ago bore the title "What the FLoC?" because we had to explain what it was about. And now the EFF has put up a test site which can be used to determine one's FLoC-age, I suppose, with the title - this is literally the title - "Am I FLoCed?"

Leo: F-L-O-C.

Steve: F-L-O-C, yes, children, A-M-I-F-L-O-C-E-D dot org, amifloced.org.

Leo: And I'm on Firefox, so I presume I am not FLoCed.

Steve: You could not be, yes.

Leo: Right. And Brave has decided not to - they're a Chromium port, but they've decided not to include FLoC.

Steve: Interesting. Well, at this juncture. But of course they're also Brave, so their whole deal is privacy.

Leo: Right, they don't want it, yeah.

Steve: But if you click that little red button there, Leo, that you had on the page a second ago - I tried to click it for you through the camera, but it didn't work.

Leo: Yeah, no FLoC. I am not FLoCed. You have to have Chrome 89 or up.

Steve: Ah, okay. So we already know from our coverage three weeks ago that the best way to characterize the EFF's position is that any form of bias in the treatment of users of the Internet, no matter the means, is a fundamentally bad idea, says the EFF, because it discriminates among Internet users who, in their view, should all be treated identically. And of course, as we know, Google's FLoC creates a temporary and transient tag formed from hashing the websites Chrome has visited during the previous week, the immediately previous week. So even though it is expressly not tracking, and even though Google plans to terminate all third-party cookie support in Chrome, which yay, we should herald that, in their transition to FLoC the EFF will be satisfied with nothing less than an entirely non-customized experience on the Internet.

So they've created Am I FLoCed to test and display whether you and your instance of Chrome may have been randomly chosen to participate in Google's initial FLoC research, which Google refers to as an "origin trial." Odd phraseology, but that's what they call it. The EFF's page opens with the headline: "Google is testing FLoC on Chrome users worldwide. Find out if you're one of them." Chances are not, but I can't wait to get some feedback from our listeners because, based on the probabilities, we'll certainly have some who are.

The EFF said: "Google is running a Chrome 'origin trial' to test out an experimental new tracking feature" - which it isn't, but okay, EFF - "called Federated Learning of Cohorts, a.k.a. 'FLoC.' According to Google, the trial currently affects 0.5% of users in selected regions, including Australia, Brazil, Canada, India, Indonesia, Japan, Mexico, New Zealand, the Philippines, and the U.S. This page will try to detect whether you've been made a guinea pig in Google's ad-tech experiment."

Okay. So 0.5% is one in every 200 Chrome users. And assuming a statistically neutral assignment, we will surely have many listeners among us. I checked two of my Chrome instances at my two locations, and I came up negative at each. So it'll be interesting. What's also interesting is the EFF's take. I'm going to share a little more of what they wrote.

They said: "What Is FLoC? Third-party cookies are the technology," they wrote, "that powers much of the surveillance-advertising," as they refer to it, "business today. But cookies are on their way out." And I would argue, yes, thanks to Google. But they said: "Cookies on their way out, and Google is trying to design a way for advertisers to keep targeting users based on their web browsing once cookies are gone. It's come up with" - "it's" meaning Google - "has come up with FLoC.

"FLoC runs in your browser. It uses your browsing history from the past week to assign you to a group of other 'similar' people around the world. Each group receives a label called a FLoC ID, which is supposed to capture meaningful information about your habits and interests. FLoC then displays this label to everyone you interact with on the web." And remember from our first discussion of this, that's one of the points EFF makes is that it's a beacon saying this is me.

Leo: Yeah, that's the big problem, I think; right? Because anybody can see it. That's, when I found that out, I thought, wow, that's bad.

Steve: That is true. And so they say: "This makes it easier to identify you with browser fingerprinting." So that's one of the other reasons EFF doesn't like it is that, as we know, fingerprinting takes advantage of as many different signals as your browser is making available. And, wow, if it's a FLoC ID it's going to be a big signal with many bits of non-entropy to help track you. So they said: "This makes it easier to identify you with browser fingerprinting and gives trackers a head start on profiling you. You can read EFF's analysis and criticisms of FLoC here." And they have a link to, yeah, the rant which we looked at a couple weeks ago.

So they said: "The Chrome origin trial for FLoC has been deployed to millions of random Chrome users without warning, much less consent. While FLoC is eventually intended to replace tracking cookies, during the trial it will give trackers access to even more information about subjects." Okay, that's true. But then cookies are going to go away, assuming this happens. They said: "The origin trial is likely to continue into July of 2021, and may eventually affect as many as 5% of Chrome users worldwide," so one in 20. And they said: "See our blog post."

So under "How can I opt out," they said: "For now, the only way for users to opt out of the FLoC trial in Chrome is by disabling third-party cookies." Which is news to me. That's interesting. They said: "This may reset your preferences on some sites and break features like single sign-on. You can also use a different browser. Other browsers, including independent platforms like Firefox, as well as Chromium-based browsers like Edge and Brave, do not currently have FLoC enabled.

"If you are a website owner" - and this is interesting. "If you are a website owner, your site will automatically be included in FLoC calculations if it accesses the FLoC API or if Chrome detects that it serves ads." Probably their ads. "You can opt out of this calculation by sending the following HTTP response header." So the header is Permissions-Policy: and then interest-cohort= and then a null set, so just open and closed parens [Permissions-Policy: interest-cohort=()]. Which is interesting.

So if you were - of course, that's on the server side. But if for some reason you didn't want visitors coming to your site, to have your site's FLoC identity, whatever that is, we'll talk about the SimHash in just a second, added into or to affect your visitors' FLoC ID, then you could arrange, you could easily add that response header to your server, your site's server responses, which would instruct the FLoC accumulating, you know, the FLoC ID accumulating browser, at this point only some instances of Chrome, to not consider that you had visited that site.

So finally, what does my FLoC ID mean? "If you have been assigned a FLoC ID, it means that your browser" - that is, if you clicked this test, right, and it says, oh, here's your ID. And I'd love to know if it changes, like every week. Anyway...

Leo: So it does tell you your cohort, so you know who you are. Yeah, they're supposed to change regularly.

Steve: Yeah, that would be interesting.

Leo: Yeah. Now I want it. I'm going to get Chrome. What cohort am I in?

Steve: Exactly. And is it right? Is it wrong?

Leo: I can't touch this. I literally don't run Chrome anywhere. Which is good.

Steve: Yeah, it is.

Leo: I'm starting to be glad.

Steve: So it means that your browser has processed your browsing history and assigned you to a group of - and they have in quotes "a few thousand," which I'm skeptical about, too. I mean, they're saying on the order of 33,000 groups. But if you multiply 33,000 by a few thousand, you get like 33 million. Well, there are a lot more than 33 million Chrome users in the world. So either it needs to be a larger FLoC ID, meaning more granular tagging, or the groups have to be larger than few thousand. Anyway, we'll see how this evolves. But something's not quite right about this.

So they said: "The numeric label is not meaningful on its own. However, large advertisers like Google and websites like Google will be able to analyze traffic from millions of users to figure out what the members of a particular FLoC have in common." So that's sort of interesting. That says that there isn't, like, bits are not assigned in the ID, meaning, oh, that bit's for commerce sites. This bit's for outdoor camping. This bit's for autos. It's not like that.

So apparently - and we'll talk about the SimHash, as I said, in a second. So it's very fuzzy. And so you need to be somebody like a Google who has its fingers out everywhere, like Analytics, right, they've got Analytics probes everywhere. Their Analytics probe will receive the FLoC ID. They will know what site their Analytics probe is on. So they'll be able to aggregate these FLoC IDs over the whole Internet, effectively, and reverse engineer that these IDs are have been seen on all of these websites where we have Google Analytics probes. And of course Facebook is the same way with their Like buttons everywhere, and advertisers serving ads across the Internet.

So you can sort of see how this closes the loop. It isn't tracking. It is profiling, which is the word I'm going to recommend in a minute that they switched to using for accuracy's sake. And it does require a lot of backend work to be constantly associating these tags, which are changing weekly. And I guess maybe the FLoC IDs themselves would settle down after a while, and then who was carrying them would change from week to week. Anyway, really interesting technology.

And so I'm going to stop sharing what they wrote. We have enough of that. So you and I, Leo, indicated three weeks ago that it seemed like an improvement over explicit tracking by third-party cookies, although when we dug into it a little bit further and realized, okay, you know, one of the EFF's points is that it does present a tag, a who-I-am-ish tag, to sites who have a way of understanding it. It may, though, be that, due to the nature, I mean, it's not going to be a tag with an obvious public meaning.

So if I saw tags coming to me from visitors at GRC, I don't have probes all over the Internet like Google and advertisers do. So it's just nonsense for me. What I did want to say is that I wish EFF would drop their insistence on calling it "tracking." It is not tracking. And I think it weakens them, their position, which has some merit, if they call it "profiling." Which, you know, is equally derogatory in this connotation.

Leo: Yeah, and maybe worse.

Steve: Yeah, people don't want to be profiled, but that's really what it's doing. It's says "profiling distillation," essentially. So what's different about this? I promised to talk about the SimHash. I was looking into it a little bit more. A highly desirable and in fact a required feature of any traditional cryptographic hash is that a tiny difference, even a difference of one bit, in what we call the digest, right, the hash's input, a one-bit change yields dramatically different hashed results. And in fact when we talked about the actual technology of cryptographic hashes in the past, we noted that, if you change one bit in a big digest that you feed to a cryptographically strong hash, on average a random set of half of the resulting bits will change. Which is mind-boggling. That's just so cool. You change one bit in what you feed into the hash, and on average a random set of half of the hash's output bits change.

But the SimHash algorithm that Google plans to use deliberately produces similar hash results - thus the name SimHash - when given similar inputs. So in other words, if you hash the big digest that we were just talking about, but didn't change much of it, the output won't change much either. That is, similar input results in a similar hash value. It's still a hash, but two inputs that may have similarities will result in a similar result, which is interesting.

So there's a sense of like a computation, a mathematical concept of distance, you know, how far away from each other are two similar things that you run through the SimHash. And it approximates that. So that's like the secret behind Chrome's ability to pour all the places you go to on the 'Net over the course of a week into this SimHash and distill it down to something which is in some way representative of the nature of the places you went without explicitly revealing where you went. And that's, from the user's standpoint, that's kind of cool because, in the same way that a password hash, where it's exacting, right, if you put the same password in, you get the same hash, which is how we use password hashes. Yet the hash itself tells you nothing about the password. Similarly, presumably the SimHash won't reveal which similar sites you went to, but that apparently you went to some.

So anyway, I guess as a tech junkie I think it's kind of cool. It'll be interesting to see how it all evolves. But in any event, I would love to have our Chrome-using listeners go to amifloced.org and see where this thing says, yeah, at the moment you are. Tweet me at @SGgrc on Twitter, if you go there and you get some positive results. I think it'd just be kind of cool.

While we're doing follow-ups on past topics, the week after our "What the FLoC" episode was "GIT Me Some PHP." And as our listeners know, that was about what I felt had to be the oh so deliberately obvious hack of the PHP project's private GIT server. The press all saw only the fact that the GIT server had been hacked and a backdoor had been installed, or code for a backdoor was there. But the way it was done, you know, the guy was sending up fireworks.

So the middle of last week we received an update on the PHP project's ongoing investigation of exactly what happened. Now we know. Nikita Popov, who is one of the two people, he was the main guy involved in tracking this down, and it was his identity along with Rasmus's, the original PHP inventor/designer, whose name was used in the two commits which created this hack. So he posted: "Update on git.php.net incident." His posting is long and detailed, and its link is here in the show notes for anyone who's curious. But the short version is that some legacy stuff bit them in the butt. They had a seldom-used secondary and less secure means for pushing commits into their private GIT repository.

Nikita, who is very much on top of this, wasn't even aware of this secondary back-channel at the time. He wrote: "When the first malicious commit was made under Rasmus's name, my initial reaction was to revert the change and revoke commit access for Rasmus's account, on the assumption that this was an individual account compromise. In hindsight," he said, "this action didn't really make sense because there was at the time no reason to believe that the push occurred through Rasmus's account in particular. Any account with access to the php-src repository could have performed the push under a false name."

Then he said: "When the second malicious commit was made under my own name, I reviewed the logs of our gitolite installation in order to determine which account was actually used to perform the push. However, while all adjacent commits were accounted for, no git-receive-pack entries for the two malicious commits were present, which means that these two commits bypassed the gitolite infrastructure entirely. This was interpreted as likely evidence of a server compromise.

"Something I was not aware of at the time is that git.php.net intentionally supported using changes not only via SSH," and he says, parens, "(using the gitolite infrastructure and public key crypto), but also via HTTPS. The latter did not use gitolite, and instead used git-http-backend behind Apache2 Digest authentication against the master.php.net user database." He says: "I'm not sure why password-based authentication was supported in the first place, as it is much less secure than public key authentication."

Anyway, he then shows us a chunk of the Apache HTTP server log showing the two commits, and he notes: "It is notable that the attacker only makes a few guesses at usernames, and successfully authenticates" - meaning has the password - "once the correct username has been found. While we don't have any specific evidence for this, a possible explanation is that the user database of master.php.net has been leaked, although it's unclear why the attacker would need to guess usernames in that case.

"The master.php.net system, which is used for authentication and various management tasks, was running" - and here it is - "very old code on a very old operating system/PHP version, so some kind of vulnerability would not be terribly surprising. We've made a number of changes to increase the security of this system." Good. So four bullet points: "Master.php.net was migrated to a new system running PHP 8 and renamed to main.php.net at the same time. Among other things, the new system supports TLS 1.2, which means you should no longer see TLS version warnings when accessing this site." So, yeah, it had been a little creaky. "The implementation has been moved towards using parameterized queries, to be more confident that SQL injections cannot occur. Passwords are now stored using bcrypt." Elsewhere he had mentioned that they were using MD5, which has long been deprecated everywhere.

And, finally, "Existing passwords were reset." So then he says where to go if you need to generate yourself a new one because your old ones won't work. So it's a little distressing that right in the middle of the home of PHP we find a very old server with, like, sending warning messages because it doesn't support any of the new TLS protocols that our browsers are using, on a very old platform, running a very old PHP that no one has looked at or is really even aware of for quite a long while. And you know me. Old doesn't automatically mean bad. Plenty of old things were written well and have stood the test of time. But big complex operating systems, feature-laden web servers, and PHP do not generally number among those things. So now we have the point of entry.

I still think of the attacker as more of a prankster due to the crying-out-loud code he placed onto the server which demanded to be noticed. He somehow arranged to authenticate as two very high-profile developers, again making sure his changes would be caught. And they never really did determine exactly how, and they really didn't care. They just got rid of it all, replacing it with up-to-date solutions to do the same thing.

And yes, they did also move the entire working PHP repository - which up to that point had just been a backup repository, now it's the main working repository - over to GitHub so they can focus upon PHP development and not worry about the security of the access controls of the repository. So all a good move. A little embarrassing. But it's nice, I mean, it's good for them to say, you know, yeah, this thing was so dusty that we're not surprised that somebody was able to crawl in. We don't really care how. We just got rid of it all.

A constant in our industry is the dilemma of deliberately terminating important critical patch support for previously supported systems that have reached the end of their support lifecycle. We often talk about this with Microsoft and Windows. It's especially irksome when some people are receiving paid-for updates while others are not. So it's not as if those updates don't exist. But even mighty Microsoft occasionally, we might even say often, bends to the severity of their own mistakes to offer out-of-lifecycle patches when the cost to them of doing so, if only I guess in reputation damage, would be prohibitive.

They just did this at the start of March by reaching way back to patch Exchange Server 2010 for the ProxyLogon flaws, even though that version was well past its kill-by date. But we've also, we've seen similar situations where Cisco, for example, and in this case, makes a different call. And it's not as if Cisco's commercial products are not similarly littered with patchable vulnerabilities. They are. Not to mention that spate of secret accounts and passwords that's appeared throughout their high-end products as people started poking around in their networking firmware and found that they'd hard-coded all those backdoors. We were covering that for a while a couple years ago.

Today, Cisco has informed the world that it has no plans to fix critical security vulnerabilities affecting some of its smaller SOHO, the small business routers. What does it tell its past customers to do? Replace them with new Cisco devices. Or perhaps it's time to consider changing brands. The bug they all share is tracked as CVE-2021-1459. And it carries the difficult-to-achieve CVSS score of 9.8 out of 10. It's hard to get up there. Just about the only way to get a higher score is if it's able to attack you after it's been unplugged. In this case, it's a bad vulnerability. Four routers are affected: the RV110W, which is a VPN firewall; and three small business routers, the RV130, RV130W, and the RV215W routers.

In each of the four cases, the known flaws allow an unauthenticated, remote attacker to execute arbitrary code on the affected device. In other words, 9.8. The flaw, which stems from improper validation of user-supplied input in the, wait for it, web-based management interface - when have we ever heard of that being a problem? - could be exploited to send specially crafted HTTP requests to the device to achieve remote code execution.

Cisco's advisory said: "A successful exploit could allow the attacker to execute arbitrary code as the root user on the underlying operating system of the affected device." They also said: "The Cisco small business RV110W, RV130, RV130W, and RV215W routers have entered the end-of-life process." Painful as it is in this case. "Customers," they said "are encouraged to migrate to the Cisco small business RV132W, RV160, and RV160W routers."

So I looked them up. The routers are their lowest end little plastic box consumer box routers. The "W" suffix in each case means "Wireless," so that's the wireless flavor. The new replacements, for example the RV160 and RV160W are $114 and $145 respectively at the moment on Amazon. So, you know, in any event, not a huge investment. It's not as if they're leaving their major enterprise customers out to dry. No self-respecting enterprise would have one of those, hopefully.

And sadly, even if they were to update the firmware, that is, if they were to offer updates of the firmware of those old and no longer being made or supported routers, how many of them would even ever get the updated code? Right? I mean, these things are forgotten in a back room. We know that there will be tons of those older routers out on the 'Net, thanklessly doing their job, day in and day out.

And presumably, people chose the Cisco brand because they'd heard of Cisco and wanted the assurance of a superior product. So let's hope that the default configuration was to disable remote web access and that no one ever had it turned on. This is another of those, we see them too often, web management interface issues. Hopefully no one is exposing their router's web management interface to the Internet. And I'm sure that all of our listeners know that unless remote management is really needed and is being actively used, web management, all remote management, should always be kept off of the public Internet.

One thing to note is that all of the tech press which covered Cisco's announcement of this trouble led their stories saying, "Cisco says," and I'm quoting one of them, "Cisco says it will not patch three small business router models and one VPN firewall device with critical vulnerabilities." That was repeated over and over in similar words. So it appears to matter to the tech press. At the same time, how long are they supposed to be responsible for routers that they are no longer selling?

I just thought I would say, so that we can sort of take this as a learning moment, the one right way to handle the need for remote access is to turn off remote access everywhere, then use one high-quality SSH server that requires a password and a certificate and a time-based additional factor for authentication. And while you're at it, run that server on some random port. There's no reason not to. Don't leave it on the default SSH port. If you use SSH to securely - oh, I'm sorry. So then you use SSH to securely first gain access to the internal network, then perform any required administrative work from the inside over LAN-bound interfaces. And it's even possible to run a standard Windows Remote Desktop connection over SSH. It works great.

So to me that's the way to do this is, you know, I've often talked about this. If your remote ends have fixed IPs and fixed IP ranges, absolutely use IP address filtering so that nobody can even see those ports when they're scanning from Shodan or any of the growing number of commercial or private scanners. Only people like at a sister IP network by IP address are able to even have any access. But if for some reason you need roaming access, where you're able to connect from any IP, the only thing, the only presence ought to be an SSH server somewhere that is high quality, that you're paying attention to. You've got three different ways to authenticate - something you know, something you have, and something that's changing all the time. You need to use those that get you in. And then from the inside you can do all kinds of things. Today that's the way to set this up.

Okay. Our final, before we get to our big conversation about Pwn2Own. I titled this "Failure to Patch." And it'll lead us to talk a little bit more about the nature of this kind of problem. Way back in early 2019, Fortinet, major Internet supplier of Internet-connected appliances, they produce among other things the FortiGate SSL VPN. Early in 2019 they received notification through responsible disclosure for a critical remotely exploitable vulnerability in several current releases of their FortiOS, as they call it, which forms the basis of several products. The security researchers in question were Meh Chang and Orange Tsai from the DEVCORE Security Research Team. We'll be hearing their name again later in this podcast.

And we've talked about Orange Tsai and DEVCORE recently. That's the guy who, or maybe it's a real name, I don't know, but in any event they're the people who informed Microsoft in 2020 that they had a problem with Exchange Server. And we're going to be talking about them in Pwn2Own. So you pay attention when these guys say, hey, we found a critical remotely exploitable vulnerability in your SSL VPN. So Fortinet shortly found and immediately fixed the trouble. They were told in early 2019. It afflicted three branches of their FortiOS, the 5.4, 5.6, and 6.0 branches.

And in May of 2019 they produced update patches, and their FortiGuard Labs, which is the security research side, published a clear vulnerability disclosure explaining that: "A path traversal vulnerability in the FortiOS SSL VPN web portal" - web portal - "may allow an unauthenticated attacker to download FortiOS system files through specially crafted HTTP resource requests." Okay, now, once again, a path traversal; right? We keep having those problems occurring also as a consequence of the fact that we have hierarchical directory structures, and ../ moves you up a level. Anyway, that was May 2019.

Three months later - so they announced it. They contacted their customers, sent out email, made the official declaration, had the patches out. The DEVCORE guys, responsible disclosure; right? They let them know. And it turns out the DEVCORE guys were going to make a presentation in the upcoming Black Hat conference. So three months later, on August 28th, Fortinet posted a blog titled "FortiOS and SSL Vulnerabilities." That was the title. And they explained that: "At the recent Black Hat 2019 conference held in Las Vegas this past August 3rd through 8th, security researchers discussed their discovery of security vulnerabilities that impacted several security vendors, including Fortinet. All of the vulnerabilities impacting Fortinet were fixed in April and May of 2019."

But in their disclosure, this disclosure in August, we get another little interesting tidbit. They said: "In addition, it was also disclosed and fixed in May that FortiOS included a 'magic'" - and they had that in quotes - "a 'magic' string value that had been previously created at the request of a customer to enable users to" - oh, this is always so bad - but "at the request of a customer to enable users to implement a password change process when said password was expiring. That function had been inadvertently bundled into the general FortiOS release, and an Improper Authorization vulnerability resulted in that value being usable on its own to remotely change the password."

Leo: Oh, god.

Steve: So basically they left a magic string...

Leo: A backdoor, basically.

Steve: Backdoor, yes, that allowed an unauthenticated - because after all, if you forgot the password, you couldn't log in to authenticate yourself, so let's just have a magic password bypass.

Leo: And it was discovered, apparently.

Steve: Yeah, exactly, yeah. Well, because it was revealed at Black Hat, which their previous disclosure did not say because it was too embarrassing. Right? It's like, okay, we'll just - we fixed it. Don't worry. Just please...

Leo: Yeah, yeah. We fixed it. Don't worry your little head about it.

Steve: Yeah, get yourself patched quickly. Oh, lord. So a few days before that, this had hit the tech press because it was, you know, Black Hat. So for example, Dan Goodin, writing for Ars Technica, titled his story "Hackers are actively trying to steal passwords from two widely used VPNs." And Dan opened with: "Hackers are actively unleashing attacks that attempt to steal encryption keys, passwords, and other sensitive data from servers that have failed to apply critical fixes for two widely used virtual private network (VPN) products, researchers said."

He said: "The vulnerabilities can be exploited by sending unpatched servers web requests that contain a special sequence of characters, researchers at the Black Hat security conference in Las Vegas said earlier this month. The pre-authorization file-reading vulnerabilities resided in the FortiGate SSL VPN installed on" - wait for it - "about 480,000 servers." So, yeah, this is a popular solution, nearly half a million servers. "And the competing Pulse Secure SSL VPN, installed on about 50,000 machines." And that's researchers from DEVCORE Security Consulting reported. So, yeah, 480,000 FortiGate SSL VPN instances. Yikes. At the time, the Internet was being sprayed - that's the term that was used in several of the other reports - sprayed with probes seeking to find and exploit these now well-known weaknesses.

Okay. So then nearly a year passes. On July 16th of 2020, Fortinet blogs with the title: "APT29 [Advanced Persistent Threat] Targeting SSL VPN Flaws." So in July last summer, 2020, they're blogging that the well-known APT29 group are targeting their devices. They wrote: "United Kingdom's National Cyber Security Centre (NCSC) and Canada's Communications Security Establishment (CSE) have published research into the activity of APT29, also known as the Dukes" - or of course we know them as Cozy Bear, they're Russians - "who have been targeting various organizations involved in COVID-19 vaccine development in Canada, the United States, and the U.K., highly likely with the intention of stealing information and intellectual property relating to the development and testing of COVID-19 vaccines." And, you know, there was reporting, right, about espionage last summer relating to COVID-19 and attempts to get in. Now we understand one of the ways in.

They wrote, Fortinet wrote: "The initial attack vectors for this group have been unpatched vulnerabilities in SSL VPN solutions from Fortinet. One of the vectors included a vulnerability resolved by Fortinet in May of 2019" - so more than a year before this - "allowed an unauthenticated attacker to download FortiOS system files through specially crafted HTTP resource requests as disclosed in" - and then they linked to their original disclosure. "At the time of the disclosure, Fortinet made available patches for all supported releases (5.4, 5.6, 6.0, 6.2).

"Customers were notified at the time via the public PSIRT advisory system of the need to upgrade immediately, and highlighted the same in the release notes. For those unable to upgrade, mitigations were provided. For additional transparency, this was again highlighted in a blog in August 2019 after the vulnerabilities were disclosed by the researchers at Black Hat 2019."

So here we are. And today, this nearly now two-year-old issue is back in the news because Kaspersky Labs has just released their report analyzing a few high-profile ransomware attacks, employing a relatively new strain of ransomware called "Cring," C-R-I-N-G, and it's well designed. It uses AES 256-bit keyed encryption. And just for the heck of it, an 8092-bit public key. So public key crypto with overkill bit length because why not. These attacks and Cring have been used to shut down some major European industrial enterprises.

Kaspersky said: "The attackers exploited the CVE-2018" - so 2018, right? - "13379 vulnerability to gain access to the enterprise's network. The vulnerability was used to extract the session file of the VPN Gateway. The session file contains valuable information, such as the username and the plaintext password." Because absolutely, let's log the plaintext password. What could be wrong with that? Unpatched FortiGate devices - on the other hand, you know, maybe they know the magic key that lets you log in without having the password, so there's that.

They said: "Unpatched FortiGate devices are vulnerable to a directory traversal attack, which allows an attacker to access system files on the FortiGate SSL VPN appliance. Specifically, an unauthenticated attacker can connect to the appliance through the Internet and remotely access the file 'sslvpn_websession,' which contains the username and password stored in cleartext. The vulnerability affects devices that run FortiOS versions 6 to 6.0.4, 5.6.3 to 5.6.7, and 5.4.6 to 5.4.12."

Several days - and this gets back to our point, Leo, about surveillance first. They said, Kaspersky said: "Several days before the start of the main attack phase, the attackers performed test connections to the VPN Gateway, apparently in order to check that the vulnerable version of the software was in use on the device. The attackers may have identified the vulnerable device themselves by scanning IP addresses. Alternatively, they may have bought" - as in purchased - "a ready-made list containing IP addresses of vulnerable FortiGate VPN Gateway devices." Because why not? They said: "In autumn 2020, an offer to buy a database of such devices appeared on a dark web forum.

"After gaining access to the first system on the enterprise network, the attackers downloaded the Mimikatz utility to that system. The utility was used to steal the account credentials of Windows users who had previously logged in to the compromised system. With the help of Mimikatz, the attackers were able to compromise the domain administrator account, after which they started distributing malware to other systems on the organization's network. They used the Cobalt Strike framework for that purpose. The Cobalt Strike module was loaded on attacked systems using PowerShell."

So they have a much greater in-depth report for anyone who's interested. I have the link in the show notes. And, you know, once upon a time, and we've talked about this before, Leo, as we old timers vividly recall, our systems, whether they were Windows, Mac, Unix, or Linux, crashed with some regularity. They don't any longer. And I'm really curious to see whether our use of networking and networking technologies will follow the same path. Are we ever going to get it right? Can we? One of the advantages our operating systems have is that everything is concentrated in one place. You know, they're self-contained monoliths of technology. And even so, at the margins, our operating systems are still far from perfect.

But on the Internet, everyone rolls their own, and everyone wants to. So as a consequence, we keep seeing the same mistakes being made over and over again. And we even see regressions where problems, once fixed, later reappear. And lessons once learned, like "don't use secret strings in firmware," are forgotten or unlearned. You know, someone new comes along and thinks, "I'm going to solve that problem this way," even when many others before have learned the hard way not to.

On the Internet there's no central control, which is often touted as being a good thing. It spurs and spawns innovation. And I'm sure that's true. But innovation brings mistakes. The old adage, "If you're not making mistakes, you're not trying hard enough," is unfortunately all too true on the Internet. But collectively we really cannot afford to keep making the same mistakes over and over. They're becoming more and more expensive as the Internet is becoming more and more core and critical. And as our look at Pwn2Own is going to show, there are undoubtedly many that have not yet been found.

In this instance it's certainly the case that many of the FortiGate SSL VPN gateways, I'm sure, 480,000 of them initially, I'm sure a bunch were quickly patched by IT professionals who were on top of their game. They received and read the news of an update, understood its significance, and quickly patched their enterprise's servers, even if they may have briefly inconvenienced some of their users in order to keep them safe. But we know that not everyone received the news; or, if they did, they received so much of it on a daily basis that they put it into the "I'll deal with this later" pile and never got back to it.

One thing we've seen is that there are technologies that seem to be particularly troublesome. Aside from embedding secret strings in firmware, there are web interfaces. They're attractive because they're so user-friendly. But they're also inherently attacker-friendly because their user-friendliness comes by way of complexity. And complexity is at an eternal war with security. So it appears that the best we can do is design our solutions to minimize our attack surfaces, like by using, as I said, one high-security SSL server to provide all remote access by moving all other remote access to the inside.

And of course, as we've often mentioned, being careful to keep all lines of communication open to all the vendors of our products and to carefully consider the implications of every security update notice we receive. Lord knows how many of those out-of-cycle, out-of-life, all-but-dead Cisco plastic box low-end routers, I mean, they're out there. They're on the 'Net. Let's hope they don't have the web interface exposed. They are discoverable. They will be discovered. And now that bad guys know there's a way in, they'll take a look at, I mean, you could buy one off eBay because, right, they're not going to have their firmware fixed. So buy one. Everyone knows there's lots of documentation on reverse-engineering SOHO router firmware. And find the problem, and wow. So the world we live in today, Leo.

Leo: All right. Let's PwnIt and OwnIt.

Steve: So last week - of course it was a virtual event, right, because we're still in COVID land.

Leo: Right, right. In a way that's better because bad guys operate virtually; right?

Steve: That's a very good point. You're right. So last week, Tuesday through Thursday, $1,210,000 flowed from some generous, deep-pocketed sponsors to some very clever security researchers during what was the Spring 2021 Pwn2Own contest. And the blow-by-blow details are as interesting as ever, and we'll get to those in a second. But overall, 23 teams of researchers targeted web browsers, virtualization offerings, servers, and what was grouped as "enterprise communications," Zoom among them.

The total prize pool, most of which was distributed, was $1.5 million, so as I said, 1.21 million went out in awards. And there was a Tesla Model 3, but none of the teams chose to tackle the Tesla this year. But as we'll see in a minute, they pretty well minced up Windows 10, Microsoft Teams, Exchange Server, Ubuntu Desktop, Chrome, Edge, Safari, and the Parallels VM Desktop. And in addition to cash, which remains kings, point scores, the Masters of Pwn point scores were also awarded for successful hacks. And the top three teams all tied at 20 points each.

Okay. So here's what happened. Starting Tuesday morning, April 6th, at 1000 hours, Jack Dates from RET2 Systems targeted Apple Safari in the Web Browser category, obviously. Jack used an integer overflow in Safari and an out-of-bands write to get kernel-level code execution. In doing so, he won $100,000 and 10 Master of Pwn points. An hour and a half later, here comes DEVCORE for the first of several. DEVCORE targets Microsoft Exchange, of course in the Server category.

Now, recall, as I reminded us already, that it was DEVCORE who originally discovered and reported the authentication bypass bugs in Exchange Server that led to this year's devastating ProxyLogon attacks, while Microsoft appears to have badly underestimated their impact, and took their time releasing patches. Well, lest we think that all of the problems with Exchange Server have been resolved, we should think again. The DEVCORE team combined an authentication bypass - yes, another one - and a local privilege escalation to completely take over Exchange Server, pocketing $200,000, and 20 Master of Pwn points.

And you know, this time it's not difficult to imagine that Microsoft will be patching Exchange Server with some alacrity. But since today, here on April 13th, is April's Patch Tuesday, those fixes may not have been ready in time. I haven't even looked yet at what is happening today underneath us. We'll check into that next week. So maybe we'll see an out-of-cycle update. Who knows? Certainly they don't want to let an authentication bypass and complete remote code execution vulnerability languish in Exchange Server because they're still being patched.

At 1300 hours the researcher who goes by "OV" targeted Microsoft Teams in the Enterprise Communications category. He combined a pair of bugs to demonstrate code execution on Microsoft Teams and, in doing so, earned himself $200,000 and 20 points towards Master of Pwn. At 1430, Team Viettel took aim at Win10, going for a Local Escalation of Privilege, and they, too, succeeded with an integer overflow in Windows 10 to escalate their privilege from regular user to system privileges, earning them $40,000 and four points towards the Master of Pwn.

At 1530 hours, the STAR Labs team consisting of Billy, Calvin, and Ramdhan targeted Parallels Desktop in Virtualization, and this brought us the first failure of the first day. They were unable to get their exploit to work within the time allotted. At 1630 hours, Ryota Shiga of Flatt Security, Inc. targeted a fully patched and up-to-date Ubuntu Desktop, hoping to achieve an escalation of local privilege. Ryota used an out-of-bands - I'm sorry, out-of-bounds, I just have OOB here - out-of-bounds access bug to elevate himself from a standard user to root on Ubuntu Desktop, earning himself a tidy 30,000 and three Master of Pwn points in his own Pwn2Own debut. He had never done this before.

At 1730, undaunted by their inability to penetrate the Parallels Desktop previously, that STAR Labs team of Billy, Calvin, and Ramdhan went after Oracle's VirtualBox.

Unfortunately, they were unable to accomplish their penetration within the allotted time, and that ended Day One.

Starting at 9:00 in the morning on Wednesday, April 7th, Jack Dates again, he returned from RET2 Systems. He kicked off the day, much as he did on the day before on Tuesday, this time targeting Parallels Desktop. Jack nailed it by combining three bugs: an uninitialized memory leak, a stack overflow, and an integer overflow to escape from the Parallels Desktop and execute code directly on the underlying OS. Jack added $40,000 to the $100,000 from the first day, racking up an additional four Master of Pwn points. So he's now at 140,000 so far.

At 1000 hours, Bruno Keith and Niklas Baumstark of Dataflow Security targeted Chrome and Edge - of course Edge uses the Chromium engine now - in the Web Browser category. They successfully employed a type mismatch bug to exploit the rendering engines in both Chrome and Edge, of course same engine, because of course we're headed toward a web browser monoculture, for better or for worse. They earned $100,000 in total and 10 Master of Pwn points.

At 1130, Team Viettel, they were back targeting Microsoft Exchange Server also, and scored a partial success, partial only due to a previous disclosure. They did successfully demonstrate their code execution on Exchange Server, but some of the bugs they used in their exploit chain had been previously reported in this contest. So probably a coincidental collision. This counts therefore as a partial win, but did award them 7.5 Master of Pwn points.

At 1300 hours, Daan Keuper and Thijs Alkemade from Computest targeted Zoom Messenger, which we'll have a little more to say about later because it was significant. This was also in the Enterprise Communications category, like Microsoft Teams. And this one made some news. They successfully used a three-bug chain to exploit Zoom Messenger and get code execution on the target system. Whoopsie. And this was without the target clicking anything. It's a zero-day, zero-click exploit which earned them $200,000, 20 Master of Pwn points, and I'm sure a follow-up conversation with Zoom.

At 1430, Tao Yan of Palo Alto Networks went after Windows 10. By using a race condition bug, Tao was able to successfully escalate his access to full system privilege on a fully patched Windows 10 machine. That earned him $40,000 and four points toward the Master of Pwn. At 1530, Sunjoo Park, who's apparently also known as Grigoritchy, was targeting Parallels Desktop. And sure enough, by using a logic bug in Parallels, he was able to execute code on the underlying OS, basically breaking out of the VM container and earning himself $40,000 and four points.

At 1630, Manfred Paul also targeted the Ubuntu Desktop and achieved or hoped to achieve local escalation of privilege by using an out-of-bounds access bug. He succeeded in escalating to root on Ubuntu Desktop. So he's now $30,000 richer and scored himself three points. At 1730, the researcher known as "z3r09" targeted Windows 10 with a local escalation of privilege attack. He successfully used an integer overflow, escalating his permissions to NT AUTHORITY\SYSTEM account, which simultaneously escalated his bank account by $40,000 and his Master of Pwnage by four.

Which brings us to the final day. Also at 900 hours, Benjamin McBride from L3Harris Trenchant also targeted Parallels. Boy, that Parallels was a popular target this year. And Ben employed a memory corruption bug to successfully execute code on the host OS from within Parallels Desktop, earning himself $40,000 and four Master of Pwn points. At 1000 hours, Steven Seeley of Source Incite thought that Microsoft Exchange probably still presented some low-hanging fruit. Although Steven did successfully use two unique bugs in his demonstration, he was only credited with a partial win because his attack required a man-in-the-middle aspect, so it wasn't solely one-sided. But it was great research, and the judge awarded him 7.5 Master of Pwn points.

At 1130, Billy, operating solely from the STAR Labs team, targeted Ubuntu Desktop. Although he was able to successfully escalate his privileges to root, the bug he used was already known to - he was targeting, oh, yeah, Ubuntu. It was already known to Ubuntu and was on their patch list. So Billy's demo of this earned him two additional Master of Pwn points, but that was it. At 1230, Fabien Perigaud of Synacktiv targeted Win10. And despite Fabien's reported excellent use of ASCII art during his demonstration - I didn't see it...

Leo: That's critical, though.

Steve: Oh, you've got to have good ASCII art, absolutely. And it turns out Microsoft was aware of the bug he used. So he did earn two Master of Pwn points for the partial win. And, yes, for the ASCII art.

Leo: It doesn't seem fair that because the company's aware of it, even if they haven't published it, that you should lose points.

Steve: I agree with you. And Leo, it also doesn't seem fair to me, if there was a way to capture the exploits, like to capture them and escrow them, it doesn't seem fair if earlier in the same competition someone uses the one that you had. You ought to share the prize money. Right?

Leo: Yeah, there you go. That would be fair, yeah.

Steve: Because the sequence is just arbitrary. Because they're sequential, and in this case on the next day, you might argue, oh, the guy gleaned something from the demo. But anyway. We have the first woman ever, Leo.

Leo: Ever? That's surprising.

Steve: Yes, at 1330 Alisa Esage went after a Parallels Desktop penetration. Despite her great demonstration and, yes, replete with ASCII art, the bug used by Alisa had been reported to ZDI prior to the contest, which as you said, Leo, I agree, shouldn't have counted against her.

Leo: I mean, if it's public, yeah. But if it's never been made public, hey, she found it on her own, independently.

Steve: Right. In this case it reduced it to a partial win. The judges commented that it was great work and were thrilled that she broke ground as the first woman to participate as an independent researcher in Pwn2Own history. She took home a pair of Master of Pwn points, and let's hope we see her again.

At 1430, Vincent Dehors of Synacktiv targeted Ubuntu Desktop. And despite Vincent's admission that this was his first exploit ever written for Linux, he had no issues escalating to root through a double-free bug, thus earning himself $30,000 and three Master of Pwn points. And Ubuntu is the wiser for it. At 1530, Da Lao took aim at Parallels Desktop and, using an out-of-bands write, he, too, successfully completed a guest-to-host escape in Parallels, earning himself $40,000 and four points toward the Master of Pwn. And at 1630, last but not least, Marcin Wiazowski targeted Windows 10 for a local escalation. He nailed it using a use-after-free flaw to escalate his Windows 10 privilege to system, taking home $40,000 and four Master of Pwn points.

So a bunch of high-profile companies have some high-profile patching to do. And every Pwn2Own, where we watch talented researchers, research hackers essentially, appear to so easily find and exploit previously unknown flaws - these are all true zero-day exploits when they're demonstrated - it always gives me a bit of a chill. And it really begs the question, just how much other unknown stuff lies out there, either undiscovered or, worse, previously discovered and being put to quiet use? We don't know what Zerodium's portfolio looks like today, and I'm not sure I want to know.

And I said I would mention a little more about these newly revealed Zoom vulnerabilities, which were demonstrated by the team from Computest Security. Theirs is particularly noteworthy because they require no interaction of the victim, or from the victim, other than being a participant on a Zoom call. What's more, it affects both Windows and Mac versions of the app, though it's not clear whether Android and iOS may also be vulnerable.

Leo: And it doesn't affect the web version of the app, either.

Steve: Oh, okay, right. Fortunately, details of the flaws have not been disclosed. But in a statement sharing the findings, Computest said that they were able to almost completely take over the system and perform actions such as turning on the camera, turning on the microphone, reading emails, checking the screen, and downloading the browser history. So it sounds more like an escape to browser sort of thing based on what they're doing, like reading emails, well, maybe Yahoo and Gmail, you know, but maybe not emails outside.

Leo: Yeah. If the browser's sandboxed, they can't get outside of that. Although it does beg, okay, now let's find a sandbox escape. We're on our way.

Steve: Right. For its part, Zoom has said that it already pushed a server-side change to patch the bugs, and noted that it's also working on incorporating extra protections to resolve some security shortcomings. A Zoom spokesperson said: "On April 9th we released a server-side update that defends against the attack demonstrated at Pwn2Own against Zoom Chat. This update does not require any action by our users." Again, that suggests, yes, it's the web only. That way, like when you bring up a new web instance, because they said it was a server-side update, right, so it's going to push out the new client-side stuff to that web chat instance.

And they said: "This update does not require any action by our users. We are continuing to work on additional mitigations to fully address the underlying issues." So that sounds like they pushed an immediate short-term, but incomplete, fix to block the explicit attack or the explicit vulnerability that had been found and demonstrated, and that it revealed some need for some deeper re-engineering, which they're going to follow up and do. Zoom said it's not aware of any evidence of active exploitation by these issues, while pointing out the flaws don't impact in-session chat in Zoom Meetings. So it's only just Zoom Chat. And they did say the attack can only be executed by an external contact that the target has previously accepted or that's part of the target's same organizational account.

So in any event, Pwn2Own is clearly providing a valuable service. And I hope that, for what it's worth, those in charge of security everywhere, you know, get the same feeling of chill that I do when they watch what talented researchers are able to do with some of the industry's presumably most secure offerings. It's like, you know, pay us enough money, and we'll take a hard look at your product and find a way in. And so you have to imagine that there are people elsewhere being motivated either by money or prestige or their government. Who knows? Anyway, we're not going to run out of content anytime soon on this podcast, Leo.

Leo: And the point of Pwn2Own, as you said, is to get these exploits into the hands of the people who can fix them, and that's where the money comes from, which is probably why something they already know about isn't worth very much because, well, we already know that. We're not going to pay you for it.

Steve: They don't want to pay for that one, yeah.

Leo: But I think it's really an important thing. And it's great that people can make a living, and some of them a very good living, if you're really skilled at it, finding these exploits. We need that. That's one of the solutions to what you were talking about, you know, can we ever live in an exploit-free world? Which is, as you say, highly unlikely, thank goodness. At least get us to Episode 999. That's all we're asking.

Steve: The world is going to keep screwing around with this stuff. And, oh.

Leo: Steve Gibson, always a pleasure. You'll find Steve's life work, SpinRite, the world's best hard drive - actually shouldn't say hard drive, any disk, SSD, too - maintenance and recovery utility. Really works with SSDs, which is such good news. That's the new SpinRite 6.1, that's going to be great. You can get SpinRite 6 right now at GRC.com. And you'll get a free upgrade to 6.1 when it's out, plus you get to participate in the development of 6.1, which is ongoing and active. Right? No honeymoon; right?

Steve: No. No, we didn't. I think we actually went back to work that evening.

Leo: Oh, my god. I'm not surprised. You've got to do what you love, man. So get that SpinRite, GRC.com. While you're there, of course, you can get a copy of this show, 16Kb. Steve has two unique versions of it, a 16Kb version for people with not a lot of bandwidth. He also has very nicely written, because it's written by a human being, Elaine Farris, transcripts, so you can read along as you listen. That's all at GRC.com. He also has a 64Kb audio version.

We have audio and video at our website, TWiT.tv/sn for Security Now!. While you're there you'll see there's a bunch of buttons at the top of the page, including the YouTube page, so you can click there and subscribe there if you want, or find a podcast application. Click one of those buttons, Apple Podcasts, Google Podcasts, et cetera, et cetera, podcasts, subscribe there. Do us a favor, leave a review if you're subscribing. Let everybody else know what a great show, a must listen, Security Now! is.

We do the show every Tuesday around about, it's roughly 1:30 Pacific, 4:30 Eastern, 20:30 UTC if you want to watch us live. There's a live audio and video stream at TWiT.tv/live. If you're watching live, chat live at irc.twit.tv. There's also asynchronous communications available. Steve has a great forum at his website, GRC.com. Is it GRC.com/forums?

Steve: Yeah, forums.grc.

Leo: Forums. Oh, the old school. That's the best way to do it, subdomain, forums.grc.com. We have our own forums at www.twit.community. We also have a Mastodon instance. It's kind of an open source federated Twitter clone at twit.social. More than a thousand TWiT listeners in there now, which is really great. We love having you on both those platforms. And if you want to follow Steve on Twitter, he's @SGgrc. His DMs are open, so you can leave questions there or at the website, GRC.com/feedback.

Might be a little late next week. I don't think so. Apple's got its event at 10:00 a.m. Pacific. MacBreak Weekly will follow immediately after. I suspect we'll be, you know, they're going to do an hour. They're very disciplined now that they prerecord these. So I think we'll not be late. But just a word of warning. Thank you, Steve. Have a wonderful evening with your honey.

Steve: Thank you, buddy.

Leo: We'll see you next week on Security Now!.

Steve: I'll be doing exactly that. Bye.

Leo: Bye.