SECURITY NOW!

Transcript of Episode #902

A Generic WAF Bypass

Description: This week we answer another collection of burning questions: Is there no honor among thieves? What was discovered during this year's Toronto Pwn2Own competition? What did we learn from last Tuesday's patch fest? Whose fault was the most recent Uber data breach? What happened when Elon tried to block all the bots? What's the first web browser to offer native support for Mastodon? What exactly is "Coordinated Inauthentic Behavior," and why is it such a problem? What will happen to GitHub submitters at the end of next year? What measure could every member of the U.S. Senate possibly agree upon? Exactly what applications are there for a zero-width space character? And finally, what larger lesson are we taught by the discovery of a serious failure to block a problem that we should never have had in the first place? The answer to all those questions and more await the listeners of today's Security Now! Podcast #902.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-902.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-902-lq.mp3

SHOW TEASE: It's time for Security Now!. Last show of the year! Steve Gibson is here. We're going to talk about the results of the Pwn2Own competition. Everybody got hacked. Everybody. Then it's the latest on last week's Microsoft Patch Tuesday. And, finally, what exactly is coordinated inauthentic behavior? You'll find out. Nothing but authentic behavior next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 902, recorded Tuesday, December 20th, 2022: A Generic WAF Bypass.

It's time for Security Now!, the show where we cover your security, your privacy, your online agenda, online with this guy right here, the king of security, Mr. Steve Gibson. Hello, Steve.

Steve Gibson: And Leo, you'll be glad to know, actually Paul Thurrott will be very glad to know that I left my Grinch costume...

Leo: You were so good. Lisa and I - so we're talking about the show that's not yet aired. It's going to be a Christmas Day version of TWiT. Steve, Jeff Jarvis, Doc Searls, Paul Thurrott and I were the old guys talking about the year's news and so forth. But Steve really dressed it up with his Grinch costume. And the thing that I was so impressed with is you didn't just do it, like, token Grinch. You kept it going the whole 2.5 hours. He was doing his hands. You were impressive. Lisa and I both were really impressed, I have to say.

Steve: Well, now everybody listening is thinking, okay, I didn't know if I was going to make time to actually watch a podcast on Christmas Day.

Leo: You have to watch this.

Steve: But maybe...

Leo: But you can watch it whenever you want. You can watch it next year.

Steve: I've got to see Gibson being the Grinch.

Leo: No, I thank you. Steve goes all in. He put his heart into that. And it's one of the reasons, you know, I think that this show does well, and you've really helped the network. I'm very grateful to you, and I always will be. Steve, we don't have a show next week. We have a Best Of. You can take next week off.

Steve: Oh, thank goodness. Oh, what did I say? No.

Leo: No, that's okay. Steve...

Steve: I get to work on SpinRite. It's going to be a major SpinRite work...

Leo: If you wanted to take six months off, it wouldn't be me that you'd have to worry about. Or Lisa.

Steve: Oh, I know.

Leo: It'd be the fans.

Steve: You can't know how many messages I receive about new ways to number the shows so that 999 is not a problem.

Leo: That's going to be a bad day in Blackrock when that happens.

Steve: Oh, it's not going to be good.

Leo: But we'll think about it.

Steve: Actually, on Episode 900, it was only two weeks ago, when I said, yeah, we have 100 episodes left, I got immediately scolded. Gibson, where is your off-by-one map. That would be 99 shows left. It's like, oh, that's...

Leo: He had an overflow, a math overflow, buffer overflow.

Steve: One has to be very careful when talking to our audience, Leo. But I think that's why they appreciate my being careful as I can be.

Leo: Absolutely. Absolutely.

Steve: This week we're going to answer another collection of burning questions. First, is there no honor among thieves? What was discovered during this year's Toronto Pwn2Own competition? What did we learn from last Tuesday's patch fest? Whose fault was the most recent Uber data breach? What happened when Elon tried to block all the bots? What's the first web browser to offer native support for Mastodon? What exactly is coordinated inauthentic behavior, and why is it such a problem? What will happen to GitHub submitters at the end of next year? What measure could every member of the U.S. Senate possibly agree upon?

Leo: Oh, god.

Steve: Exactly what applications are there for a zero-width space character? And, finally, what larger lesson are we taught by the discovery of a serious failure to block a problem that we should have never had in the first place? The answer to all those questions and more await the listeners of today's Security Now! Podcast 902.

Leo: See, now, this is exactly why you've got to do more than 999 shows. You've come up with the perfect way to introduce the tease to the show. It's perfect. All those questions and more will be answered. I like it.

Steve: You know, and my theory - it matches my theory of life. Just as you're taking your final dying last gasp of a breath, you figure it out.

Leo: Oh. I thought you were going to say, you say "Is that all there is?"

Steve: It's like, oh, I finally get it.

Leo: Oh, I get it. Eureka.

Steve: And of course the kids never want to listen to Gramps. He has no idea what he's talking about.

Leo: No. Yeah.

Steve: So I just keep it to myself.

Leo: We yell at the cloud so you don't have to.

Steve: That's right.

Leo: We'll take a little break, and then we're going to get the answers to all those questions and more with Steve Gibson.

Steve: And of course I forgot to mention that the title of today's podcast is...

Leo: Oh, what is it?

Steve: It's "A Generic WAF Bypass."

Leo: Is that the answer to any of those questions?

Steve: Happy Holidays.

Leo: A Generic WAF Bypass. All right.

Steve: A Generic WAF Bypass is involved with the larger lesson that we're taught by the discovery of a serious failure to block a problem that we should have never had in the first place.

Leo: So that's what I thought. Okay. So that last one...

Steve: That's right.

Leo: ...is the laugh. Okay.

Steve: Boy, what a - yes.

Leo: I don't even...

Steve: We're going to have a buildup to that one.

Leo: That's why you listen to this show; right, folks? That's exactly right. I am ready for the Picture of the Week.

Steve: And this one...

Leo: Ooh.

Steve: ...really is interesting.

Leo: Yes.

Steve: So I titled this "Old School Message Routing." And it'd be difficult to adequately describe this, but it's a fascinating picture. And if nothing else, Leo, it shows us how far we've come because what we have is...

Leo: I'm old enough to remember going to a department store, buying something with my mom, and the clerk would roll up the slip, put it in a tube...

Steve: Like a cylinder.

Leo: A cylinder, and shove it into one of these holes. That's a pneumatic delivery system.

Steve: And my earliest memory of that was when you did a car, you know, auto ATMs. I remember that the very earliest...

Leo: Yeah. Oh, yeah, we still have one in town, yeah.

Steve: Yeah, where you'd actually, like, stick your checks and things in this plastic cylinder and then stick it in this tube, and it would go [sucking sound].

Leo: It was so satisfying.

Steve: Oh, it was wonderful. So, okay. So what we have, you know, if people could imagine a bunch of tubes that have sort of like a catcher at the end, so that this cylinder is going to come flying out of this tube and, like, stop. This is the switch room, which involves all of the ends of these tubes, and some poor guy who's standing there, I guess like picking tubes up from one, like picking cylinders up that have arrived in one tube and then sticking them up in another tube and off they go. You know, on to their destination. This looks like a message routing or switching room for pneumatic tube transfers. And but now what's interesting is that I don't see any labels on these things. There's like, I don't know, what, 30 or them or 20 of them that we can see. And then over on the left are a whole bunch, like other rows of them. This is just fascinating.

Leo: This is the inbound. This is your inbox. I don't know what you would do for outbound.

Steve: Yeah.

Leo: I guess you have to remember where it came out of. These are called, according to the chatroom, Lamson tubes. And there's a website dedicated to it, pneumatic.tube.

Steve: Oh, that's so good. That's so good. And also I love the one there along the ceiling at the top. Like it looks like it starts to come down, and then it changes its mind. It goes, oh, nope, we're not going to end here. We're going to go across and go back up somewhere else. So it's like, okay. And also when you think about it, you know, these things have some length to them; right?

Leo: Oh, yeah.

Steve: Like it's actually a canister that you're able to put documents in. So there's a minimum radius for the bend of these things, or the canister's going to get stuck trying to go around a curve. So it must be that it had larger ends and a thinner body, or maybe even like a concave body that would allow this thing to navigate around a corner because it's got - we can see some corners. They're not sharp. And they couldn't be. Anyway, just, you know, completely not about packet, well, it is kind of about packet routing, I guess.

Leo: Oh, yeah. This is a website all about it. Here's a Scientific American article about the pneumatic tube system of New York City. Now, it was published a couple of years ago. But the original article was from 1897.

Steve: This is why the Internet is so good.

Leo: There's nothing you can't find. I mean, pneumatic.tube. There's a website, for crying out loud.

Steve: Love it.

Leo: Amazing. Amazing.

Steve: I love it.

Leo: Yeah. And they're all over the world, I guess, these pneumatic - there's a pneumatic railway. I'm not sure I'd want to ride that.

Steve: Just lie flat and tuck your arms in, and we're just going to close this little lid on this round coffin and send you on your way.

Leo: Yeah. I think actually that's - isn't that Elon Musk's idea for - what did he call that, that tube system that he wants? Not the boring company, but he actually wanted to do a high-speed tube.

Steve: Hyperloop; right?

Leo: Hyperloop, thank you.

Steve: That was last week. Who knows what tomorrow will bring.

Leo: He's a little busy now, yeah.

Steve: Yeah. So a malware operation known as URSNIF, which we've noted a few times, they've kind of crossed our radar, it's the fourth one this year to suffer from internal squabbles which end up surfacing in the public eye. Disagreements over Russia's invasion of Ukraine and in some cases strong pro-Russian sentiments which have divided previous groups were the nominal triggers in those first three instances. But for number four it appears that it's just about greed. You know, that's all the motivation we need. And this is heading toward the answer to the question, you know, is there no honor among thieves.

Through a Twitter account which was @URSNIFleak, which may still be suspended, it was a while ago at least, an ex-member of this group, this URSNIF group, announced his intention to leak the real world identities of the top leaders of the group unless he received a significant payout. To prove his willingness to do so, in a succession of tweets @URSNIFleak leaked various pieces of internal dialogue, some of the group's source code, and the names of three low-level group members. That was enough to get this person paid. After that he tweeted, he said: "I just made more money in a single week than I have made in years. Pay workers right, and they won't have a reason to leak stuff."

And I couldn't help but note that it's interesting that this person considers this had been an act of "making money." Wow, what a different culture. Apparently, the motivation to extort was heightened by something that the head of the group said in an interview with the VX-Underground project, though it's not clear what about that was upsetting. The URSNIF leaker tweeted: "The interview angered me. He has been a bad boss for a long time. I've been waiting for the right time to release." And of course, remember that this is a bad boss of a Russian ransomware group. So, yeah, "bad" kind of goes with the territory.

Leo: That's hysterical. I just wanted to collect my paycheck and go home, but no.

Steve: Yeah. So I had to extort money in order to get paid, or extort the boss for money. That's right.

So anyway, I don't think we can count on all of the major groups to implode, but there's probably a little extra tendency for that to happen within an organization comprised of people who must be aware that what they're doing is not earning an honest day's living. At least I guess we can hope so.

Okay. Pwn2Own Toronto 2022 just happened. And it's always interesting to see what hackers wearing white hats are able to do to today's fully patched and up-to-date systems, right, because those are the targets is in every case these things are 100% patched. And we've seen instances in the past where a group will get all ready to demonstrate a vulnerability that they've very cleverly crafted in something and, like, the day before their demo the publisher patches. And not because they told them; right? I mean, they will end up telling them. All of the things that are done during these Pwn2Own contests end up being communicated to the publisher of the thing that was compromised, but not beforehand. Anyway, so the point is, this is state-of-the-art, fully patched as good as we know how to make it, products that these guys are going after.

So in the past we've taken people through a blow-by-blow. And sometimes I think that ends up getting a little long. So I'm going to summarize this a bit. The recently concluded Toronto 2022 hacking contest focused upon hacking routers, smartphones, printers, and other smart devices. So it was sort of an IoT-esque, you know, smartphones, printers, routers, and other stuff. It was a four-day contest that ended up getting won by DEVCORE, which is the now well-known Chinese/Taiwanese penetration testing group.

Okay. So to give everyone some sense for this, I'm just going to quickly scan down, and I abbreviated these, just the bullet points which briefly describe the attacks. So, and this is just day one; okay? Day one of the four-day contest: A stack buffer overflow attack against the Canon imageCLASS MF743Cdw printer. A two-bug authentication bypass and command injection attack against the WAN interface of a TP-Link AX1800 router. A command injection attack which caused a Lexmark MC3224i printer to serenade the audience with a well-known Mario Brothers tune.

We had a command injection attack against the WAN interface of the Synology RT6600ax router. A stack-based buffer overflow against an HP Color LaserJet Pro M479fdw printer. An improper input validation attack against the Samsung Galaxy S22. A command injection root shell attack against the LAN interface of the Synology RT6600ax router again. Another improper input validation attack against the Samsung Galaxy S22. A two-bug attack, SQL injection and command injection, against the LAN interface of the Netgear RAX30 AX2400 router. A SQL injection on a router, that's interesting.

Anyway, two different stack-based buffer overflow attacks against the MikroTik router and a Canon printer. Three bugs, two missing auth for critical function and an auth bypass attack, against the Synology DiskStation DS920+ NAS. Two bugs, including a command injection, in an attack against the HP Color LaserJet Pro M479fdw printer. Five different bugs leveraged in an attack against the LAN interface of the Netgear RAX30, again, AX2400 router. And three different bugs against a Netgear router and an HP printer. Now you know why I'm only doing day one. And remember, these were all 100% up-to-date devices, all cut through. All of that on only the first day. And it kept going like that throughout the entire event.

As we know, LAN-side attacks on routers and NAS devices are much less concerning than attacks that can be launched against the WAN interface. But this contest revealed plenty of both of those. And the number of printer vulnerabilities that still exist, well, I suppose we shouldn't be surprised. But obtaining well-hidden persistence inside a network is an overriding goal of anyone who penetrates an enterprise's perimeter. And printer protocols by their design loudly broadcast and advertise on networks because their goal is to be found. Unfortunately, this results in highly vulnerable printers shouting their presence and creating a perfect and often unsuspected place for malicious post-intrusion malware to set up shop and wait, thus becoming an advanced persistent threat.

So anyway, I just, sort of as a reality check, here's, yes, these guys are at the top of their game; right? They're the world's best hackers. Yet it appears that all they have to do is look at some device, make that the target of their scrutiny, and they can find a way in. So, you know, we need to - I guess anyone listening to this podcast long enough will have lost any sense that there's anything that's invulnerable to somebody who is serious about finding a way in. And in fact that is the story behind today's main podcast story at the end.

Okay. And speaking of getting into networks, it's not just lower-end IoT devices that are permitting bad guys to get into networks. Both Citrix and Fortinet, who are two of today's largest providers of enterprise networking equipment, recently released security updates to patch zero-day vulnerabilities, one in each of their devices, that were being exploited in the wild against them. In the case of the Fortinet zero-day, which created an unauthenticated remote code execution in the FortiOS, which is what runs the company's SSL-VPN devices, it was the way some ransomware was managing to crawl inside enterprise networks. Which is never what you want. And it was so bad that Fortinet did the right thing by also offering down-version patches for their older, out-of-support devices which were still running their also-vulnerable 6.0 firmware.

The zero-day was first spotted being used in the wild by a French security firm Olympe last week, and to Fortinet's credit they patched it over the weekend in just three days. So props for getting it fixed quickly. But, boy, you know, what this French security firm watched were ransomware groups gaining entry to an enterprise network through this vulnerability. So, wow.

And I said, you know, two zero-days, one each, Fortinet and Citrix. Citrix's is the other. And it's also an unauthenticated remote code execution exploit. Interestingly, this one was spotted by the NSA - yeah, our National Security Agency. In their security advisory, the NSA wrote that they saw the Chinese cyberespionage group designated APT5 leveraging that Citrix zero-day, but the NSA offered nothing further about what was being done with the obtained leverage. So again, high-end gear also vulnerable, not just low-end stuff.

Last Tuesday was the industry's increasingly well-attended final monthly patch event of the year. And those offering up incrementally more secure improvements in their code and products notably included Adobe, Android, Apple, Microsoft, Mozilla, SAP, and VMware. Microsoft fixed 72 security flaws this month across their range of offerings, and that included a zero-day that was being used to circumvent Microsoft's SmartScreen and Mark-of-the-Web detection which was - I got myself tangled up. The zero-day was being used to bypass that to allow standalone JavaScript files to execute because modern Windows will execute JavaScript natively. And of course we covered this trouble recently, so it's very good that it's been fixed.

The other issue Microsoft addressed was a problem that we also noted before here, which was that somehow malicious Windows drivers were being used by the Hive and the Cuba ransomware strains or groups, and those malicious drivers were being trusted by Windows because they were carrying valid Microsoft signatures. Whoops. Okay. In this month's advisory, Microsoft wrote: "We were notified of this activity by SentinelOne, Mandiant, and Sophos" - so everybody was watching - "on October 19th of 2022, and subsequently performed an investigation into this activity." And I should just mention that SentinelOne, Mandiant, and Sophos, they've all got clients, and their technology is on those clients' networks offering protection over and above what Microsoft is providing.

So the reason all three of those companies all notified Microsoft on October 19th of 2022 is that's when all three of their technologies' alarms went off when drivers were acting maliciously. They immediately thought, wait a minute, how is a driver getting into the kernel and acting this way? So they yanked those, looked at them, found that they were all validly signed by Microsoft, and immediately notified Microsoft that that's what was happening. So that's the good thing about the way this industry is evolving with third parties who are offering real-time detection services for people's networks is they're able to close the loop and let Microsoft know when something bad has happened.

Microsoft said: "This investigation revealed that several developer accounts for the Microsoft Partner Center were engaged in submitting malicious drivers to obtain a Microsoft signature." In other words, there were some bad partners there. They said: "A new attempt at submitting a malicious driver for signing on September 29th, 2022, led to the suspension of the sellers' accounts in early October." So, okay. So early October, yet the drivers appeared on the 19th of October, which suggests that drivers were signed. Microsoft caught this happening at the end of September, yet there were still drivers out there that Microsoft wasn't aware because they hadn't invalidated them. So then they appeared in use toward the end of October on the 19th, and that's when they got notified of something that basically they already knew about.

Anyway, that was all good. And not to be left out, Apple also updated WebKit to fix a zero-day that was being used in targeted attacks against iOS users.

Uber has been having a rough time recently. Recall that about four months ago the Lapsus$ gang breached Uber's security and caused them trouble. What's interesting about last week's second breach, which resulted in unfortunately the leaking of the personal details of more than 77,000 Uber employees, and also some source code and credentials for some of the company's internal IT network. And I should mention Uber confirmed the authenticity of that leaked data. What's interesting is that this wasn't directly Uber's fault. The breach occurred in the network of an Uber-contracted IT service provider whose name suggests, or suggested to me at least, that all the good names were already taken. This company chose to name itself Teqtivity. It's T-E-Q-T-I-V-I-T-Y.

Anyway, the day after Uber outed Teqtivity as being the actual proximate cause of this latest leak, Teqtivity themselves disclosed the breach last Thursday. Uber may have been Teqtivity's biggest customer. Actually, I did some looking around, and they've got a bunch of them. But I mention this because other notable companies may also have had their data stolen since a breach of one large service provider can potentially expose the data belonging to all their clients. We saw this of course a couple years ago when all of those dental offices were in trouble because they were all outsourcing their dental records management to one single provider, the so-called MSP; right? Managed Service Provider.

As an industry these days, we're really sort of facing a conundrum. Do you run your own in-house shop where you are solely responsible for your company's security and IT and everything? Or do you decide that running networks and servers and points of presence and dealing with the constant need to focus upon security is not your mainline business? It isn't what you should be spending your cycles on. And also that it's just become too complicated to do it right. You know, that's a valid consideration. So you farm it out to someone who promises to you that it will be their mainline business because that's all they're going to do. It is their business.

And, you know, I think today that's a tough call. I think it can work out and be extremely cost effective to do this subcontracting so long as everything goes well. On the other hand, when something doesn't go well, you know, if it's a big breach at a major service provider, potentially the damage can be huge because so many individual clients of theirs can be affected by a single attack. So again, a tough call; but increasingly I can see that it makes sense. And this sort of goes back to the comment I made, I think it was last week, where the guys at the DigiCert customer advisory board meeting looked at me like I was nuts for still doing it myself, saying, "Gibson, nobody does their own hardware anymore."

Okay. I don't know, Leo, if every podcast on TWiT mentions Twitter and Elon. Probably. But he keeps doing things that are interesting, certainly for us.

Leo: That's one way to put it.

Steve: From the outside looking in, it's difficult to understand the mechanisms at play inside Elon Musk's Twitter reign. From the outside, anyone would get the sense of things lurching back and forth inside Twitter, presumably as Elon's, as he described it himself, his "biological neural net fires off whimsical edicts," which Twitter's remaining employees apparently quickly implement without any buffering in a desperate effort to hold onto their own paychecks in this chaotic and fragile work environment which has been created.

You know. One moment we're done with layoffs. Then we have more layoffs. No, now we're really done with layoffs. Then entire departments disappear. Collections of press accounts are suspended for an interval of seven days, until the next day they're reinstated. A new policy states that anyone tweeting a link which points to another social network will have their account suspended, until a few hours later when that policy ends. You know, it really has been quite something to watch. And as I'm assembling the notes and details of this podcast, when I follow links to online events that would once have linked to Twitter, I'm increasingly being taken to Mastodon.

Well, last week something else happened as a result of an apparent misfiring of Elon's biological neural net. He decided that he was going to block all of the bots. This, of course, was something that had endlessly bedeviled all of the pre-Elon Twitter engineers, how to block the bots. Elon, it turns out, had the answer.

So he declared publicly that he had a surprise for all of the bot farms, and last Monday Twitter blocked entire IP address blocks which were used by, it turns out, approximately 30, three zero, mobile carriers across Asia. According to Platformer...

Leo: [Making noises]

Steve: I know, this included the primary telecom providers for all of India and all of Russia, as well as Indonesia's second-largest telecom. Of course, there were vastly more legitimate users in every one of those address blocks than there were bots. So three countries' worth of legitimate Twitter users, who all shared the same IP address blocks as a few bots, were completely cut off from Twitter. And you have to, like, wonder, how could anyone not anticipate that happening? It's - I don't know. Again, it's just incredible to me. What I can see is that Elon wants to own Twitter, but Twitter is not technology. It is enabled by technology. Twitter is a community. A community can be enabled and nurtured and encouraged. The one thing it cannot be is owned. Nobody owns Twitter's community. No one can. Not even Elon.

Leo: You're always welcome over at TWiT Social. You could have your own Mastodon account. I promise not to ban you.

Steve: Well, we're going to see because, you know, he famously held a poll over the weekend.

Leo: Oh, that's all silly, silly.

Steve: I know. He said if this poll says I should no longer be CEO, I will resign.

Leo: Yeah. It did.

Steve: Of course the poll said please resign.

Leo: We're waiting.

Steve: Well, yeah, go. We're done. We're done.

Leo: Well, and then he said be careful what you wish for, which is probably true because god knows who would take over from him.

Steve: And the problem is I think he's destroyed. I mean, you know, one of the things that I'm seeing, in fact we're going to get to this in a minute, this is the "coordinated inauthentic behavior," which is just this wonderful phrase.

Leo: I love that phrase, yeah.

Steve: It is difficult to do this, Leo. It is difficult to be in an ownership or, you know, catbird position with any large social media network. You are going to be constantly fighting abuse. On the one hand, you want to open your gate and allow everybody in the world to come in and participate. Unfortunately, we know that the world has a whole bunch of bad people in it. And, you know, bots are a thing. And so it's just this is really hard to do. And I would argue Twitter was doing the best job they could. And of course then they got all - they ran afoul of all of these issues of, well, you know, should we allow people to scream fire in a burning building or not? And ad infinitum. Anyway, Elon appears to have badly broken it. And it's not all clear to me that him disappearing is going to suddenly fix it. I don't know how you do that.

Leo: Yeah.

Steve: It's sad. Anyway, the good news is, speaking of Mastodon, Vivaldi recently became the first browser to have its own Mastodon instance, Vivaldi Social. Now, the new version on the desktop is also the first to integrate Mastodon natively into the browser itself, along with the ability to pin tab groups and other UI improvements. They said: "We believe in providing alternatives to Big Tech while putting your privacy first, and launched Vivaldi Social, our Mastodon instance. And today we are integrating Vivaldi Social into the sidebar of our desktop browser, becoming the first browser to offer this functionality." So anyway, I just wanted to give a tip to Vivaldi and note that it's interesting that this has happened.

On the topic of governments recognizing the growing dangers of known vulnerabilities in the networks of the enterprises within their own borders, remember we've talked about a couple governments, I don't think it was the Dutch government, and I meant to go find out which one we'd referred to before. But there was another note of some government that, like, announced they were going to start scanning their own citizens. It might have been the U.K.

Anyway, in this case the Dutch government has been doing it. And they just said that since the beginning of this work, which was the summer of 2021, so about a year and a half ago, and about a year and a half worth of this, they have sent more than 5,200 warnings to Dutch companies concerning security vulnerabilities within their networks. Officials said that around 76%, so three out of four, of these warnings were for sensitive systems being accessible via the Internet: RDP, SMB, LDAP, and so forth. The other 24% of the warnings regarded malware infections, leaked credentials, or unpatched systems. So presumably they're scanning the Internet and seeing a version number in the greeting of something and saying, whoops, that's not the latest version. And they send the company a note saying, hey, you know, maybe you ought to update your email because you're running an old one which has some known vulnerabilities.

So anyway, this is not the first time we've encountered this, and it seems like an entirely sane thing for governments to do in the interest of helping to protect their national interests and those of all of their citizens and the enterprise operating within their borders. So I expect that we're going to be seeing more announcements of this sort in coming years.

Okay. CIB, that's the abbreviation for Coordinated Inauthentic Behavior, a term that I love. A recent report from Facebook's parent company, Meta, introduced me to this term, Coordinated Inauthentic Behavior. And I love it because it's such a wonderfully neutral and politically correct term to describe the behavior of organizations and countries that have figured out that they can use fraudulent postings and replies on Facebook to influence beliefs and behavior through massive coordinated campaigns. Facebook's report, which they published on Thursday, last Thursday, was titled "Recapping Our 2022 Coordinated Inauthentic Behavior Enforcements."

They noted that since they began focusing upon the explicit abuse of Facebook services for what they term "covert influence operations," they've disrupted 200 identifiably separate global networks that were the source of these campaigns. Those networks were based in 68 countries - but far from evenly, as we'll see - and operated in at least 42 different languages. Two thirds of the campaigns, I thought this was really interesting, two thirds of the campaigns were targeting their own local audiences in their home countries, and only one third were aimed at audiences outside the country, so abroad.

In terms of targets, more than 100 different countries, from A through Z, Afghanistan through Zimbabwe, have been targeted by at least one CIB network, foreign or domestic, with the U.S. being the most targeted with 34 of those operations; followed by Ukraine, and I'm sure that's only in the most recent year, targeted by 20 CIB networks; and then the U.K. targeted by 16. So 34 for the U.S., 20 for Ukraine, 16 for the U.K. And a single covert network might often be simultaneously targeting multiple countries at once. In one case, for example, a network running from Iran was simultaneously targeting 18 countries on four different continents.

Okay. As for the originators of the campaign networks, perhaps not surprisingly, Russia leads the list of the originating sources of these networks having 34 networks identified, closely followed by Iran with 29. And then the next highest, with fewer than half of Iran's 29, interestingly, was Mexico, which surprised me as the third largest source of these influence networks at 13. Interestingly, those are the top three, right, Russia, Iran, Mexico. China is not among them. Russia and Iran are the biggest perps in this game.

And as I said, I was surprised about Mexico, so I went looking for some more information about them. As I suspected, most of the CIB networks originating in Mexico have focused primarily on regional or local audiences to Mexico, often in the context of regional elections. Those networks tended to be less tactically sophisticated, and many were linked to PR or marketing firms, including instances where - I love this - one network simultaneously supported rivals in the same electoral post. The report noted that this illustrates the danger of using covert "Influence Operations for Hire" that might be providing inauthentic support to, not just the highest bidder, but to multiple bidders at once. So again, we have this wonderful term "Coordinated Inauthentic Behavior." And now we have some sense, thanks to Facebook's work on this, about the spread and nature of these networks.

Okay, SHA-1. We might say that we hardly knew ye, but as it turns out we knew ye quite well. The NIST has formally announced what many of us have been assuming for some time. The aging original SHA-1 cryptographic hashing function is officially being retired. In its place is either SHA-2 or SHA-3, both which have existed for quite a while and have been in use for a long time. But I did a bit of a double take when I saw that companies now have, as of the NIST's announcement, companies have until the end of 2030, in other words until the beginning of 2031, so another entire eight years from now, to make that replacement. The end of NIST's announcement said:

"Modules that still use SHA-1 after 2030 will not be permitted for purchase by the federal government. Companies have eight years to submit updated modules that no longer use SHA-1. Because there's often a backlog of submissions before a deadline, we recommend that developers submit their updated modules well in advance, so that CMVP has time to respond."

Okay. Now, a cryptographer might have been a bit more explicit and careful in the wording of that mandate. I'd have written "Modules that are still capable of using SHA-1 after 2030," dot dot dot. The reason for the added clarity is that, as we've often talked about, many cryptographic systems obtain robust interoperability by comparing acceptable protocol suites that both ends understand and then negotiating the best and hopefully the most secure among those. But through the years of this podcast we've examined a great many "downgrade attacks" where a malicious endpoint identifies that the other end is still offering a no-longer-considered-safe weak cryptographic protocol. So the sneaky end pretends that it cannot use any of the stronger systems, thus tricking the agreeable other endpoint into establishing a potentially vulnerable connection. So what we want is for all systems to immediately eliminate SHA-1 from their collection of acceptable hashing functions. Absolutely it should no longer be offered.

And, you know, it is a fine point, but for the record there are still some things you could use SHA-1 for safely if you chose. It would make a fine hash for use in a PBKDF (password-based key derivation function) where a hash is iterated a great many times. But given that its presence might allow its misuse, removing it altogether would be best.

Okay. And one last little tidbit for any of our listeners who are using WordPress. Last Wednesday WordFence, the very useful third-party WordPress web application firewall people who have been identifying troubled WordPress add-ons, they launched a free and very useful-looking vulnerability database for WordPress add-ons. I poked around it a bit, and I'm impressed. So I wanted to give our WordPress users a heads-up about it. It is at wordfence.com/threat-intel. Again, www.wordfence.com/threat-intel. Looks like a very comprehensive listing of dangerous add-ons for WordPress. I would say worth taking a look and making sure that you're not using any of those and might be unaware of the problems.

And Leo, time for me to catch up on my caffeine at the moment.

Leo: Ketchup on your caffeine? That doesn't sound very tasty.

Steve: Catch up on my - and it really is actually very tasty. I'm loving it.

Leo: Oh, good. We will not be here next week. We're going to be doing reruns. Well, we like to call them the "Best Of" shows. We carefully edit them, craft them, to be the best material from the year 2022. That'll be a week from today, December 27th. And then Steve and I will be back with a live show in two weeks, January 3rd. This is our last show of 2022; January 3rd the first show of the brand new year. Steve, let's continue on. Soldier on, as you...

Steve: And you know, you're right, Leo, ketchup on my caffeine.

Leo: You got it now.

Steve: I did. Took me a little while. I got it.

Leo: It wasn't a very good joke.

Steve: I also don't put ketchup on my eggs, for what it's worth. That's just not where...

Leo: Oh. How about hot sauce?

Steve: Oh, yeah. A little Sriracha maybe, or some Tapatio.

Leo: Yeah. Now you're talking. See? See?

Steve: There we go. Okay. So a bit of Closing the Loop feedback from our listeners. Michael Lawley, he said: "Please @SGgrc, it's pronounced meddy bank." And, okay, I'm glad to know that.

Leo: We didn't know. How would you know, med-i-bank vs. meddy.

Steve: Right, Medibank and Meddy Bank. And, you know, that does sound more Australian, doesn't it.

Leo: Meddy Bank, yeah.

Steve: Yeah, in that kind of an accent. So thank you Michael. Glad to know. SKYNET tweeted me: "Question about ADP," referring to Apple's new encryption that was the topic of last week's podcast. He says, or asks: "Once it's turned on, and the keys are sent down to your device, is it stored in hardware or software? Because what happens when you get a new iPhone in the future, how do you get the keys over to your new iPhone? You can't set up the new phone and restore an iCloud backup once you log on, so it would have to be by the method where you move your old phone close to your new phone; correct? Thanks."

Okay. And I received a number of questions that are sort of related to this. The primary concept that I guess I want to get across is similar to the familiar pattern that LastPass and presumably all other password managers use, at least I hope they would. In all of those cases, they are simply storing an encrypted blob on our behalf. They have no visibility into the blob. But by making that blob available across devices, devices are able to share a common set of passwords or, as in the case of Apple, a common set of decryption keys.

So the process of Apple relinquishing the keys to iCloud is that Apple sends the current Keychain blob, which it is never and has never been able to decrypt, and the current iCloud keys, which until now it has held in its data centers' HSMs, to the user's device. The device uses its local private account key, which never leaves the device, to decrypt the Keychain blob on the device, and then adds the current iCloud key into the Keychain. In this way, the keys that Apple was holding are moved where Apple could get at them, into the user's account Keychain, where Apple can never get at them. The device then instructs Apple to delete the iCloud keys that it just sent from all of its data centers' HSMs. Now, only the device has the old iCloud keys in its Keychain. Then, wanting to be thorough, the device performs a key rotation, changing the key that encrypts the iCloud data to one that Apple has never had in its possession.

But again, since we're all quite familiar with the notion of Trust No One and Pre-Internet Encryption, which is the technology that all password managers holding encrypted blobs that they're unable to decrypt use, I think that's the clearest way to think about this, and the best analogy. Basically, Apple is holding this stuff for us, provides the synchronization service among devices, but is only able to hand the devices these blobs which are then decrypted locally on the device in order to give devices the keys which it's then able to use to go further. You know, one of the things that I've been saying for years is that we've got all these very cool crypto components which we can assemble in any manner of different ways.

Walt Stoneburner said: "Steve, you have warned several times that pixelation is not a safe redaction technique. Someone just wrote a beautiful GitHub project that visually brings home the point," he said, "as you see unredaction being performed." And it's funny, I don't know why this started circulating again. I got a whole bunch of tweets about it. And I thought, oh, cool, something new. But it was 10 months ago when we first talked about this and showed this. So not something, turns out, that was new.

Michael Brodsted said: "Hi, Steve. Love your show. Read this article and thought it might be interesting for you." Okay. So what Michael sent, and I appreciated it, was The Verge's follow-up on their story about those Eufy cameras that we talked about a few weeks ago. Remember, those were the cameras that promised that all of their storage was local, and that nothing ever left the user's home, and that it was all transmitted directly to their phone. Then of course in some reporting, following up on some news that that was not the case, The Verge was able to monitor their own Eufy cameras from the other side of the country.

And Leo, you and I talked about it at the time. This was the company that was owned by Anker, and it was our conjecture that, you know, the way this could happen, because we like Anker, and we thought they were a reputable company, was that after having launched their successful power supply product line, they perhaps purchased the Eufy camera line in order to grow their business. Anyway, we don't know. But I guess I'd want to forgive them a little bit for making such a mess.

Anyway, The Verge checked back, and when did they find? Their follow-up story is headlined "Anker's Eufy deleted these 10 privacy promises instead of answering our questions." And the subhead reads "Two weeks after getting caught lying to The Verge, Anker still hasn't sent us any answers about its security cameras. Instead, it's nerfed the Eufy 'privacy commitment.'"

So one of the things on The Verge's page, they have this wonderful mouse-based sliding divider where you can slide the divider with your mouse back and forth, and it reveals either - it's like a shutter, revealing either the old or the new privacy claims. And if you pull it to the - I think you pull it to the right. You see the original claims where nothing leaves your facility, it's all kept locally, blah blah blah. You pull it to the other side, and you get then the updated claims which are dramatically toned down. So anyway, The Verge makes a very good point. And it's sad. But on the other hand, all of these systems are out there. They can't change the way they operate. And I'm sure they never operated the way they said they did. Someone just got a little maybe overenthusiastic or carried away when they were writing the marketing material for this. Or, you know, maybe they did add features later where like they began to offer cloud things and never updated the page in order to make it correct. So they've done that now.

Alain, he tweeted, @Alain_Gyger, he said: "Thanks for another excellent episode. I do have one question about TikTok. Do you see a difference between the bans on ZTE and Huawei versus TikTok?" He said: "The FCC has labeled all three as 'unacceptable risk.'" He said: "Also, I just saw that there is a bipartisan bill that would 'end all commercial operations of TikTok in the U.S. and other social media platforms that are sufficiently controlled or influenced by America's foreign adversaries, including China, Russia, and Iran.'" He said: "It'll be interesting to see where this goes."

So, okay. I wanted to include Alain's tweet to give me the opportunity to note that last Wednesday the entire U.S. Senate voted unanimously, passing a bill which would bar the installation of TikTok from any government-owned devices. So, yes, whereas initially a handful of Republican governors and an attorney general may have been first during the previous week or two, now we have unanimous and obviously completely bipartisan agreement on this, which is astonishing to me. Wow. But it happened.

But to Alain's question, I do regard these selective bans such as on ZTE and Huawei, and even on TikTok, as mostly ridiculous theater because we are so intimately, deeply, and inexorably enmeshed with Chinese technology products. You know, I look around, and everything in my home, all of the electronics that I own, and the electronics in what I drive, was fabricated in China. Every bit of it. And I'm sure that's the case for all the people listening to this podcast. And speaking of listening to this podcast, this podcast is literally brought to your ears thanks to networking chips and processors and transistors all made in China by Chinese citizens, and much of it was designed there.

So to me, none of this posturing and saber-rattling makes any sense. It must be that some sort of geopolitical kabuki is transpiring at a level that's far above my pay grade. I'm just a simple technologist who does understand networking and processors and transistors, so I know that if China did actually want to be evil, the West would be in deep trouble because in the interest of economy we've allowed ourselves to become utterly dependent upon products which we need from China. I don't want that to be a bad thing. I hope it's never going to be a bad thing. But if it is going to be a bad thing, then the problem is way bigger than a couple of wayward Chinese companies.

So maybe I don't get it, but it's sort of similar to me being amazed that Russia was still using Windows, like for the last many years, and still is. They've finally said that they're thinking about moving to something Linux-based, presumably. And there have been some rumblings of the same thing from China. Just to me it seems crazy that that would be the case. But here we are, utterly dependent upon another country and now saying we don't trust them. Well, we're unable not to trust them, frankly. So that's what I think.

And finally, David Ruggles. He said: "Reaching out regarding the zero-width space mentioned in Security Now! last week," he said, "I use it to fix stupid programs. For example, if you want to reference an account on Twitter without tagging them, enter the at sign, then a zero-width space, and then the account name, and it won't get tagged." And then he gave some examples of things that were not tagged, were and weren't tagged, in his tweet to me. And so he said @TheRealRuggles vs. @TheRealRuggles. They looked identical. Only one was lit up. He said: "Similarly in Excel it defaults to adding a hyperlink when you enter anything that looks like a URL or email address. You can use the zero-width space to prevent that behavior without changing the look of the text."

So that was cool. I think that's very clever. The problem is, I mean, and I should say I can see many applications for that, as well, but it leaves the question, how do you enter a zero-width space through the keyboard? I asked the Google, and I was told: "The zero-width space is a Unicode character U+200B," which is also HTML right? And Google said: "It's remarkably hard to type. On Windows you can type ALT and then 8203." Well, I tried that, and I got, what is that, the symbol for maleness, I think. Anyway, that didn't work, ALT+8203. So if anyone can figure out how to type these, how to enter the zero-width character through our keyboard, I think that seems like a useful thing to be able to do.

Leo: People were also using it to post their Mastodon link on Twitter because it didn't look the same. I mean, it looked the same, but...

Steve: Without triggering the thou shalt not post...

Leo: Yeah, their filters, yeah.

Steve: Oh, cool. Okay. Briefly I'll note that SpinRite is looking quite good. By the end of this past weekend we were at the eighth alpha release, every known weird data recovery behavior that we were seeing appears to have been resolved, and SpinRite is now cruising through even the most damaged and troubled drives. While my focus was on getting SpinRite to properly perform those primary functions, I had also been accumulating a list of less critical but still necessary to-do items. And our testers that have been getting a little restless have been suggesting new features that they'd like to have. You know, nothing big, but there are some convenient things that make sense.

So by the end of the day, Sunday, two days ago, I told the group that I would be retrenching now and disappearing for a while, while I worked my way through everything that was on the wish list and the things to be fixed. After today's podcast, that's what I'll be doing. When I return with alpha release nine it should be very close to finished. I'm sure there'll be a few loose odds and ends. That's the nature of such a complex project. But I have to say with some pride and relief that everybody who has been testing 6.1 has been very impressed by this new SpinRite's speed and capabilities. So we're getting there. It's not going to be a Christmas present. It's not going to be a New Year's present. But it's going to be early in 2023 that we finally have this 6.1 for everybody.

Okay. A Generic WAF - that's W-A-F - Bypass. As an industry, we've matured to the point where vulnerabilities are being discovered only in specific implementations of some specific solution, and only typically in specific versions of those implementations. In other words, whereas once upon a time the entire industry would realize that an established standard could be abused in an unexpected way, and everyone's implementation would need to be changed. That's where we were. A perfect example was everyone's DNS servers which were emitting queries from ports sequentially assigned by their underlying operating system and often emitting those queries with sequential identifiers. When that came to light, those who were focused on DNS realized that this would allow for successful DNS spoofing at scale, and the entire industry repaired DNS overnight.

These events stand out because, thankfully, they've become so rare. These days, as we know, problems have generally become much more obscure and specific. For example, it might be that if you're still using the out-of-support version 2.029.472 of JimmyCrack's Query Reflector, you need to update it to at least version 2.426.327 in order to avoid problems with query reflection back flush, and you should do so immediately. Those are the kinds of things we're often seeing now.

Leo: It's not real, folks. Don't go looking for Jimmy's back crack.

Steve: Oh, don't worry. Unless you actually do have JimmyCrack's Query Reflector, in which case you've got other problems.

Today's rarity of big generic protection bypasses has made their existence extremely interesting. And a group known as Team82 recently discovered just such an industry-wide mistake. They discovered an attack technique that acts as the first generic bypass of multiple web application firewalls being sold by industry-leading vendors including at least Palo Alto Networks, F5, Amazon Web Services, Cloudflare, and Imperva.

Okay. So before we proceed, we need to briefly revisit another one of those "Holy Crap!" events which hit the entire industry many years ago, and which due to its difficulty the industry continues to grapple with, and that's SQL injection. Stated succinctly, SQL injection can occur when there is some way for user-provided input to be passed to a SQL database for its interpretation. A SQL database is driven by strings of characters which express commands and queries. Simply by typing commands, new database tables can be created. They can be populated with data, queried for their data, and deleted when they're no longer needed. New users can be instantiated, passwords can be changed, privileges can be granted, all through simple text commands. And further increasing the system's power, the simplicity of this interface allows SQL databases to be queried over networks. The simplicity and the power of this interface explains SQL's success.

But the simplicity and power of this interface has also been at the heart of one of SQL's longest running vulnerabilities. Wikipedia tells us that the first known public discussion of SQL injection appeared around 1998, and cites an article in Phrack (P-H-R-A-C-K) Magazine, long since discontinued. SQL injection has been the bugaboo of web applications from the start. The first web apps gleefully presented a form asking their user to please enter their full name to look up their record in the site's database. The designer of this form assumed that that's what anyone would do. So whatever string they provided as their name would be added into a SQL query string to access the site's database.

And all was well until it occurred to some clever individual that the website had inadvertently given them direct access to that site's SQL database back end. Rather than simply inputting their name, they could, for example, input a string which closed the open query and started another entirely separate SQL command of their choosing. This allowed a remote visitor to directly issue SQL commands to the site's database. If the web designer had assumed that no one else could ever access the database, which of course is what they assumed, the SQL account behind the website's form might even have admin rights. This would allow remote visitors to do anything they might wish.

This has been such a common and persistent problem because the fundamental architecture of this system is fragile. It is not inherently secure and resilient. It is inherently insecure. We need to take user-supplied input, like some personal details, and embed them into a database query so that we can look up their record. We have to do that; right? The trouble with SQL is its power. That same query channel is also SQL's command and control channel.

This has been such a longstanding and well-understood problem that it found its way into one of XKCD's brilliant comics, and we've talked about it in the past. The first frame shows somebody holding a cup of coffee, listening over the phone. This is Mom, who's received a phone call. And over the phone she hears: "Hi. This is your son's school. We're having some computer trouble." Mom replies: "Oh, dear. Did he break something?" And we hear the voice of the distraught principle saying: "In a way. Did you really name your son Robert '); DROP TABLE Students;-- ?" And Mom says: "Oh, yes. Little Bobby Tables, we call him." And then the principal says: "Well, we've lost this year's student records. I hope you're happy." To which Mom replies: "And I hope you've learned to sanitize your database inputs."

Leo: Such a classic. Such a great comic.

Steve: Perfect.

Leo: Yeah.

Steve: And so what XKCD is telling us is like exactly this. And so here's, I mean, stepping back from this a bit, the biggest problem is, through all these years, since it was first understood near the birth of the web, no one has fixed this. Instead, we just keep patching it. We focus upon each mistake in isolation, rather than recognizing that the entire architecture is wrong for this application. SQL was not created for the web. No one would have done that. It was first designed in the early 1970s. Leo, we were just talking about when we graduated from high school.

Leo: Yeah.

Steve: Yeah, back then.

Leo: Yeah.

Steve: Yeah. That's when this, you know, IBM came up with this before there was an Internet or websites or web apps. Unfortunately, the web found it, and it's been a troubled marriage ever since. The problem is every newly created web app creates another new opportunity to make a mistake in the parsing of user-supplied input that would give a remote attacker access to the site's backend database. That's why I say that the systems we've built around this architecture are inherently brittle and fragile. That's why, still today, SQL injection attack scans are constantly sweeping the Internet looking for that newly created, newly vulnerable web app, and SQL injection remains at the top of the OWASP Top 10 list of web application vulnerabilities.

So what do we do if there's no sign that we're going to fix the underlying problem? Well, the universal solution to protecting our networks from external hostility is to place a firewall in front of those networks and force all external traffic to be inspected and to pass through that gauntlet before it's permitted to reach our interior, presumably vulnerable networks. And thus was born the idea of the Web Application Firewall, or WAF for short.

The fundamental concept of a web application firewall is detailed traffic inspection. Whereas packet-level firewalls generally look no deeper than packet headers, which specify the source and destination IPs and ports for the purpose of monitoring packet flows, a web application firewall examines in detail the content of all web application traffic transiting its boundary in order to detect and block malicious attacks.

So in XKCD's example above, a WAF would spot and block a form's input field data that contained suspicious characters for a user's name such as closed parentheses and semicolons, so that they would go no further. With a web application firewall positioned upstream of an organization's web application servers, that malicious data and intent would never reach any web applications that might not be adequately providing for their own protection. Again, you wouldn't need this if mistakes still weren't being made freshly, but they are because this is all being done wrong. But it's what we've got.

So, okay. With this background, here's what Team82 had to say about their recent discovery. They wrote: "Web application firewalls (WAFs) are designed to safeguard web-based applications and APIs from malicious external HTTPS traffic, most notably cross-site scripting and SQL injection attacks that just don't seem to drop off the security radar." Gee, imagine that. I wonder why.

They said: "While recognized and relatively simple to remedy, SQL injection in particular is a constant among the output of automated code scans, and a regular feature on industry lists of top vulnerabilities, including the OWASP Top 10. The introduction of WAFs in the early 2000s" - okay, note that time, note that date - "early 2000s was largely a counter to these coding errors. WAFs are now a key line of defense in securing organizational information stored in a database that can be reached through a web application. WAFs are also increasingly used to protect cloud-based management platforms that oversee connected embedded devices such as routers and access points. An attacker able to bypass the traffic scanning and blocking capabilities of WAFs often has a direct line to sensitive business and consumer customer information. Such bypasses, thankfully, have been infrequent, and one-offs targeting a particular vendor's implementation.

"Today, Team82 introduces an attack technique that acts as the first generic bypass of multiple web application firewalls sold by industry-leading vendors. Our bypass works on web application firewalls sold by five leading vendors: Palo Alto Networks, F5, Amazon Web Services, Cloudflare, and Imperva. All of the affected vendors acknowledged Team82's disclosure and implemented fixes to their products' SQL inspection processes. Our technique relies first on understanding how WAFs identify and flag SQL syntax as malicious, and then finding SQL syntax the WAF is blind to.

"This turned out to be JSON, JavaScript Object Notation. JSON," they write, "is a standard file and data exchange format, and is commonly used when data is sent from a server to a web app. JSON support was introduced in SQL databases going back almost 10 years. Modern database engines today support JSON syntax by default, including basic searches and modifications, as well as a range of JSON functions and structures.

"While JSON support is the norm among database engines, the same cannot be said for WAFs. Vendors have been slow to add JSON support, which allowed us to craft new SQL injection payloads that include JSON, and that completely bypassed the security WAFs provide. Attackers using this novel technique could access a backend database and use additional vulnerabilities and exploits to exfiltrate information via either direct access to the server or over the cloud. This is especially important for OT and IoT platforms that have moved to cloud-based management and monitoring systems. WAFs offer a promise of additional security from the cloud. An attacker able to bypass these protections has expansive access to systems."

Okay. So what happened? History has shown that no one is able to always get SQL injection protection correct because it's so much easier for it not to be correct. So the notion of a web application firewall is created to move the burden from individual input forms, fields, and web apps to the perimeter, where a single comprehensive web application firewall will be able to protect all of an organization's applications at once. That happened about 20 years ago, in the early 2000s. Now, remember, though, only for those organizations that deploy them. A web application firewall is like your big iron box. It's expensive. It needs to be constantly maintained. It needs to be licensed. Smaller organizations aren't going to have them. But the big guys do for the last 20 years.

The problem, of course, is that now it's become less imperative for those individual web applications, which are now safely ensconced behind their protective application barrier, to be quite so worried about their own input form field content. After all, there's a big mean web application firewall at the front gate that's going to keep little Bobby Drop Tables safely out of reach. So all is well.

But then a decade passes, and a particular syntax for describing the features and details of objects becomes popular. It outgrows its own modest origins and is adopted by other languages and applications because it does the one thing it was designed to do cleanly, minimally, and efficiently. And so the JavaScript Object Notation, JSON, grows increasingly prevalent. Perhaps it was inevitable that SQL databases would eventually choose to add their own support for JSON. And they did. Here's what Team82 had to say about that.

They said: "In modern times, JSON has become one of the predominant forms of data storage and transfer. In order to support JSON syntax and allow developers to interact with data in similar ways to how they interact with it in other applications, JSON support was needed in SQL. Currently, all major relational database engines support native JSON syntax by default. This includes Microsoft SQL, PostgreSQL, SQLite, and MySQL. Furthermore, in the latest versions, all database engines enable JSON syntax by default, meaning it is prevalent in most database setups today.

"Developers have chosen to use JSON features within SQL databases since it became available for a number of reasons, starting with better performance and efficiency. Since many backends already work with JSON data, performing all data manipulation and transition on the SQL engine itself reduces the number of database calls needed. Furthermore, if the database can work with the JSON data format, which the backend API most likely uses as well, less data pre- and post-processing is required, allowing the application to use it immediately, without the need to convert it first.

"By using JSON in SQL, an application can fetch data, combine multiple sources from within the database, perform data modification and transform it to JSON format, all within the SQL API. Then the application can receive the JSON-formatted data and work with it immediately, without processing the data again. While each database chose a different implementation and JSON parser, each supports a different range of JSON functions and operators. Also, they all support the JSON data type and basic JSON searches and modifications."

And here's the key underlying what Team82 discovered: "Even though," they wrote, "all database engines added support for JSON, not all security tools added support for this comparatively new, though decade-old feature, which was added as early as 2012. This lack of support in the security tools" - meaning the WAFs - "introduced a mismatch in parsing primitives between the security tool - the WAF - and the actual database engine which is implementing SQL, and caused SQL syntax misidentification."

They said: "From our understanding of how a WAF could flag requests as malicious, we concluded that we needed to find SQL syntax the WAF would not understand. If we could supply a SQL payload that the WAF would not recognize as SQL, but the database engine would parse, we could actually achieve the bypass. As it turns out, JSON was exactly this mismatch between the WAF's parser and the database engine. When we passed valid SQL statements that used the less prevalent JSON syntax, the WAFs did not flag requests as malicious.

"The JSON operator '@>' which checks whether the right JSON is contained in the left one, threw the WAFs into loops and allowed us to supply malicious SQL payloads and allowed us to bypass the WAFs. By simply prepending simple JSON syntax to the start of the request, we were able to exfiltrate sensitive information over the cloud."

So this forms a very interesting story. We start with a fundamentally insecure design when a powerful database system from the '70s which was never designed to allow malicious users to access its command input stream is used as the backend database for websites, thus inadvertently giving malicious users access to its command input stream. Rather than recognizing that using SQL in this way is fundamentally a horrific mistake, every individual website must patch their input field parsers in an attempt to prevent SQL command and query syntax from being submitted by the visitors to every site. SQL injection becomes a meme, and XKCD captures its essence.

In an extension of the firewall concept, web application firewalls are created to centralize and concentrate the SQL syntax filtering challenge. And all seems fine for a time. Then SQL syntax undergoes a fundamental extension as all SQL servers implement support for the increasingly popular JavaScript Object Notation. But despite this extension, some of the industry's web application firewalls fail to update their protection logic to incorporate an awareness that JSON can now be used to encapsulate and issue SQL queries. Fortunately, a team of white hat security researchers stumble upon this tidbit while they're working to discover just such a bypass, and they quietly inform the many vendors of those vulnerable web application firewalls of their discovery. And all is well again.

Or is it? Because SQL is still powering virtually all web applications, and the fundamental problem of now an even more powerful SQL syntax existing still remains. If JSON could be used to slip past web application firewalls to reach the SQL database behind, how many websites, individual websites that are not being protected by a big iron web application firewall, might now be vulnerable today to exactly the same JSON bypass? Happy New Year.

Leo: A lot of the apps that I've used either use MySQL, which isn't the - it's SQL syntax.

Steve: That's it, MySQL.

Leo: So it counts.

Steve: No, it's exactly the same. It supports JSON.

Leo: Yeah, as long as it supports JSON. How about SQLite? Same thing?

Steve: Yup.

Leo: Okay.

Steve: SQLite, MySQL, PostgreSQL, and MS SQL.

Leo: As long as it supports the SQL language, which that's the IBM, is the SQL Server. That's the original.

Steve: Right. Right.

Leo: But these all support that language. So they're all...

Steve: Yeah, and even MariaDB is a...

Leo: Really. Maria also supports it.

Steve: Yeah.

Leo: Yeah, because everybody knows SQL and knows that language; right?

Steve: Yup.

Leo: So why would you get a new one?

Steve: And unfortunately, exactly, why would you invent a new one? It's horrible to use it as a backend for the web. But it's the one we've got.

Leo: Well, I mean, you could use it as a backend. You just don't want to expose it.

Steve: The problem is, if you say, you know, look up the username that the user inputs, you're taking the - you're inherently taking the string they gave you.

Leo: And that's going to be a SQL string, yeah.

Steve: And inserting it into a query.

Leo: Yeah, yeah.

Steve: I mean, so the problem is that that query is not just a query. It's also command and control, account creation, table deletion, I mean, it is - it was never meant to be exposed to arbitrary input. But, oh, look, we got SQL. Let's use it as our backend.

Leo: And sanitizing your inputs merely, I mean, it requires you to be clever enough to catch all the...

Steve: Perfect. Perfect.

Leo: You have to be perfect.

Steve: Perfect every single time.

Leo: Yeah, yeah.

Steve: That's why I say this is inherently broken, inherently bad.

Leo: Right, right. What would be the alternative? A new language? Well, any database query language is going to have - is going to be prone to this problem; right?

Steve: Well, no, because a database query language should not let you delete the database that you're querying. That's not a query language. That's a command and control language.

Leo: Right. So separating the queries from the control and command would be the solution.

Steve: Yes. And, you know, it's reminiscent of the printf we talked about Apple getting tripped over, where the problem was they - the printf inherently mixes control with text.

Leo: Right, right.

Steve: And that's a bad idea.

Leo: You should see what the format string in Lisp can do. I mean, it's printf on steroids. It predates printf because it's Lisp.

Steve: Right.

Leo: And it is crazy the things you can do with that. It's a programming language in and of itself.

Steve: Right.

Leo: And that's probably not a good thing. I would never - I could never imagine opening your website to a random format string. So I guess I can see the inherent problem here, yeah.

Steve: Yeah. You know, I don't know how we fix it. You could preserve, you could reengineer it so that the query was fundamentally limited through that channel, you know, so that you could...

Leo: Yeah. You could say I won't accept commands of any kind, only search queries.

Steve: Right. And...

Leo: And I guess that's what sanitizing your inputs means. But it's hard to do that perfectly, especially since they're probably using regular expressions to parse it or something. I don't know. I wonder what the current best practices is.

Steve: Well, and that's just it. The problem is, you know, how many times, Leo, have we encountered, for example, a TCP/IP stack where security researchers figured out, ooh, you know, we can't do things this way. We have to do them that way.

Leo: Right.

Steve: You know, a classic example is packet fragmentation. It turns out it's bizarrely difficult to deal with fragmented packets. Yet new people come along and reimplement the TCP/IP stack and make the same errors that we fixed 30 years ago all over again.

Leo: Yeah.

Steve: Because there are some things that are just hard to get right. And the problem is, like SQL is what everyone uses as their backend, and it's a bad idea.

Leo: Well, you need a database of some kind. I think the bad idea is to allow commands to - you would think the permission structure would say, look, unless you're logged in as a permissioned user, you shouldn't be able to execute commands. And then just keep the privilege level of the web server and the web queries low. Seems like that would be solvable.

Steve: Unfortunately, the programmers who put this together never think that.,

Leo: Well, and they also want the power.

Steve: They're like, well, you know...

Leo: They want to be able to do it.

Steve: I developed this code. It's great.

Leo: Yeah, yeah. Yeah, I don't think it's insoluble. And you do need a database on the backend. I mean, that's the modern web. You don't want flat files. I bet you're all flat files, though. You don't have a database on your backend; do you?

Steve: Actually, one of the things that has been really heartening is that when you go to GRC, and you put your SpinRite serial number in, in order to get a link for the pre-release, it's shocking how fast it is.

Leo: Did you write the program yourself? Of course you did. In assembler? Of course you did.

Steve: It's a super lean embedded database that's being accessed in assembler, of course.

Leo: Yeah.

Steve: And it's just - it's a simple indexed database, and it is like, it is amazing. And it took me a while to realize everything else I use, I click the button, and it's like, okay, wait a minute, you know, it's still spinning. And then it comes up. But not GRC, it's just pow.

Leo: Nice. Very interesting. Of course this is why you have to listen to the show; right? The best, most interesting stuff. It's been a great year, Steve. I think you'll enjoy the Best Of. We found some really fun bits to put together.

Steve: Oh, neat.

Leo: That's next Tuesday. And then the following Tuesday, January 3rd, we're back again with Episode, what'll that be, 903. Yeah.

Steve: 903, baby.

Leo: Wow. We're getting close to the end.

Steve: Don't say that. Don't say that.

Leo: It's like walking off a cliff.

Steve: We've got a whole two years.

Leo: We've got a long walk off a short pier ahead of us.

Steve: That's right.

Leo: As my father used to say.

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

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