IoT Flash Botnets

Description: This week we discuss some ROCA fallout specifics, an example of PRNG misuse, the Kaspersky Lab controversy, a DNS security initiative for Android, another compromised download occurrence, a browser-based cryptocurrency miner for us to play with (and Google considering blocking them natively), other new protections coming to Chrome, an update on Marcus Hutchins, Microsoft's "TruePlay" being added to the Win10 Fall Creators Update, and some interesting "loopback" from our terrific listeners. Then we take a closer look at the rapidly growing threat of IoT-based "Flash Botnets."

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-634.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-634-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here, and we are going to talk about something kind of creepy. Somebody is building a bot army using compromised routers. It's already two million strong and growing 10,000 a day. But what are they doing with it? What are the plans? No one knows. Plus we'll talk about Google's Advanced Protection and how it works. I got my keys. I tried it. I turned it off. I'll tell you why, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 634, recorded Tuesday, October 24th, 2017: IoT Flash Botnets.

It's time for Security Now!, the show where we cover your safety and security online with my good friend, my compadre, our comrade in security arms, Mr. Steven "Live Long and Prosper" Gibson. Hi, Steve.

Steve Gibson: Leo, great to be with you again, as always, for Episode 634 of this weekly security podcast. We're recording this on October 24th. And something has happened in the last week which is - the right word would be "nascent." The title of this podcast is "IoT Flash Botnets" because almost exactly on the anniversary, the first anniversary of the Mirai botnet, something much worse has been detected.

Leo: Oh, no.

Steve: By several security firms.

Leo: So this is in the wild.

Steve: This is. And, okay, I'll give our listeners a tease. We're going to cover this at the end. Mirai brought down DynDNS, a major DNS provider, using 100,000 compromised cameras and DVRs.

Leo: Right.

Steve: This botnet, which goes by two names, either "IoT Reaper" or "IoTroop," I-O-T-R-O-O-P, currently has two million.

Leo: Oh, boy.

Steve: So 20 times the client strength, the attacker strength that Mirai had. Anyway, we'll talk about it at the end of the show. But, I mean, it could do anything it wanted to.

Leo: What can it do? Anything it wants.

Steve: Anything it wants, yes.

Leo: Uh-oh. Oh, boy.

Steve: So before that we're going to talk about some of the ROCA fallout specifics. Remember that we opened the topic last week. That was the defective Infineon library in embedded devices that has been there for a long time. And since that had just happened, all we could cover last Tuesday was the fact of it. We now are beginning to get a little more fallout specifics from that. There's a real interesting instance, thanks to Matthew Green and two other researchers, about an example of pseudorandom number generator misuse which is really interesting.

Leo: Oh, I saw that. That was very interesting, yeah.

Steve: Yeah. We need to talk about the Kaspersky Lab AV controversy. I've sort of been ignoring it because it just seemed purely political. But Eugene has responded in an interesting way that I want to talk about. Also there's an interesting DNS security initiative which the XDA Developers have picked up on for Android. Another compromised popular software download occurrence, which paints now another meme, essentially, that we're going to talk about because first we had Handbrake, then we had CCleaner, now we have - god, I can't remember. It's in my notes. It's a Mac media player, Elmedia or Eltima. Anyway, it's in the notes. So, but this is beginning to form a trend that we need to identify.

Oh, I also found this so cool cryptocurrency browser-based miner, bitcoin mining -

actually it's not bitcoin, but it's one of the cryptocurrencies - that we can play with. And when you do it, you're donating to a cause of your choice that they offer. But it also shows the hash rate of your machine. And I tweeted this earlier, and my Twitter followers have been having a ball comparing which browser is faster mining than the other, and which platforms mine at which speeds. But then of course the other thing it lets us do is experiment with the mining blockers because now we have a benign way to play with mining on our machine. So we're going to talk about that. Oh, and the fact that Google, for their Chrome browser, is now considering natively blocking this. So you could test that, too. Also there are some other new protections coming to Chrome.

We've got a quick update on Marcus Hutchins and what's going on with him. Something that just sort of caught me by surprise because Microsoft has nothing better to do than to block people from messing with gaming under Windows 10. And I'm sure this is something that Paul would be all tuned up on, and probably you are, too. But then as I read into it a little bit further, I thought, okay, well, that kind of makes sense. So it's something called "TruePlay" coming for the Fall Creators Update that has a security angle, of course.

We've got some interesting loopback feedback from our terrific listeners. And then we're going to take a closer look at this new, really worrisome threat of a flash botnet, that is, the emergence and the emerging threat of flash botnets, this one in particular that already is 20 times the strength of Mirai. And it hasn't misbehaved yet, but that's what it's for.

Leo: We're keeping an eye on it just in case.

Steve: I don't think we're going to have to keep an eye on it. We're going to know when this thing gets targeted, yes.

Leo: It's like a small child as we're waiting for it. Any minute now it's going to live up to its potential.

Steve: Hold onto the chandelier.

Leo: Well, as always, this will be a jam-packed episode, full of goodness and delight, thanks to Mr. Gibson here. Somebody asked me something this week. I wonder what you would say. Is it getting worse or better with security?

Steve: I would say it's getting worse. I think that, well...

Leo: I said both.

Steve: It's getting more.

Leo: It's more.

Steve: So it's more. So you've got more researchers finding more problems. There are more problems to find. There are more devices connected to the Internet, and there is more pressure on developers to push stuff out so there are more mistakes being made. So it's just more of everything.

Leo: And then the other side of it is, which I brought up, is because these are searched for more, there's bug bounties and all sorts of things, so I think there are more security researchers looking for flaws to protect you than ever before and more patches than ever before and more regular patches. More attention's paid. I feel like, for instance, Windows 10 is more secure than it's been ever. Things like that. So it's both, in a way; right?

Steve: Right. Well, and browsers are getting better.

Leo: Browsers are getting better, yeah.

Steve: We no longer have, beyond belief, JavaScript running in Outlook that used to be able to infect you. No more active desktop.

Leo: Right, right. So we're making progress.

Steve: Yes. Flash is being phased out finally. So, I mean, we're moving forward. And, for example, IE got rewritten to make Edge, which is arguably now state-of-the-art secure. Firefox is catching up. Chrome is leading. So there's definitely progress. But we're not really seeing anybody winning. That is, it's just more.

Leo: Yeah, that's true. Yeah, it's back and forth.

Steve: Yeah. So there's more attention being paid. What that's revealing is more problems. But it's not like there aren't new problems that are coming along just as quickly to fill the gaps left by the old problems that have been fixed.

Leo: True, true.

Steve: So, yeah, I don't think we're ever going to wrap this up.

Leo: We're in a little rat race.

Steve: We'll just do the all-day Tuesday podcast here before long.

Leo: It's not getting shorter, that's for sure.

Steve: And we'll just - maybe bring your sleeping bags, and we'll just get a fresh pot of coffee. So our Picture of the Week is textual, which we've never done before, but I was sent this screenshot, which I tweaked and cleaned up a little bit. I just sort of liked it because anyone who's ever read an RFC - and this also put me in mind, we were talking about the Wi-Fi Alliance and how their specs are secret. And I gave the example last week of the IETF and how easy it is to take for granted, but how cool it has always been that all the specs that underlie the operation of the Internet are all developed in public, in public sight.

Leo: Well, except for the IEEE and the KRACK thing; right? It's behind a paywall.

Steve: But that's not. That's not the IETF. No, but that's not the Internet.

Leo: Ah. That's true. It's a standard. It's a different standard. Yeah, yeah, yeah.

Steve: Right, right. So the Wi-Fi Alliance, they hide their crap. Bluetooth hides their crap. I mean, those are all organizations where you pay a lot of money to be on the inside. But as a consequence mistakes happen.

Leo: People make the argument KRACK could have been discovered 10 years ago.

Steve: Yes.

Leo: If the IETF wouldn't have put the spec behind a paywall, a very expensive paywall.

Steve: Well, not the IETF. That was the Wi-Fi Alliance.

Leo: Oh, okay.

Steve: And their effort. And the IEEE and the Wi-Fi Alliance, so that's different completely than the IETF, which is the Internet Engineering Task Force. So literally the name of the documents, RFC, stands for Request For Comment, which that acronym says, "We want to know what you think." And throughout the last 12.5 years we've been talking about RFCs. I'm referring to them all the time. I wrote my own IP stack, my TCP/IP stack, using these specifications. I mean, they're complete. They're thorough. They're well conceived.

At one point the structure was unified, and they came up with these - and it's always in all caps. They'll say such and such and such must be done or must not be done or should be done or should not be done. I mean, and so they formalized what is important and what is optional and what is recommended. Oh, there is also "may," so it may do this or may do that. But they're very explicit in, like, ranking both on the positive and negative end of the scale, where specific requirements fall.

And that's why this particular Picture of the Week caught me because it's titled "So, pretty much this is how you break a widely implemented protocol." It's five steps. Step 1, read the RFC. Every time it says "must," check if they did. Every time it says "must not," check if they did not. Every time it says "should," assume they did not and test for it. And, finally, every time it mentions a requirement that does not affect the functionality, assume it was done wrong by at least one company, and nobody noticed because it still works.

So, I mean, and this encapsulates so much, and sort of broadly and definitionally, so much of what we talk about every week because there are, well, in fact one of our stories about the misuse of a pseudorandom number generator specifically said this should not be done, or these numbers produced by this should not be used for these purposes, only for these. And people ignored that. And as a consequence, it's possible to look at some of the handshake traffic on very enterprise-scale VPNs and crack them because, whoops, "should" and "should not" were not honored.

So anyway, this was just a great little observation that I liked, and we'd never really talked about the structure of RFCs and, like, not only how significant it is. I would argue, I mean, we've often talked about how surprisingly robust the Internet is, from day one. I mean, consider it largely hasn't changed. And while, yes, we know it is far from perfect, our requirements today were not the requirements then. So it's not fair for us to impose how we wish it had been designed 20 years ago on the way we wish it were today because they had no idea. I mean, this was sort of an experiment back then.

But its foundation was so rigorously thought out. I mean, and that's how you get something that survives this well is that you use committees, small teams of smart people who argue with each other in email forums and back and forth, and they receive a response, they think about it overnight, and then they come up with something better. So each spec ratchets forward slowly until it finally drops out of that process as an RFC that is then adopted by the Internet Engineering Taskforce, and that becomes a spec. And if it was crap, we'd have a system that we couldn't, I mean, it wouldn't be today's Internet. But it was almost immaculately conceived. It's just phenomenal.

Leo: It was IEEE that has it behind a paywall, not IETF.

Steve: Correct.

Leo: I meant to say IEEE, yeah. For KRACK, yeah. So it's not so different. I mean, admittedly, all these companies are heavily influenced by their paying members, i.e., the big companies.

Steve: I just think, because they're asking their - the only thing I can figure...

Leo: No reason for IEEE to hide this behind a paywall. That's crazy.

Steve: Correct. Correct. I mean, it's WiFi. And I just think it's because they charge a lot of money, and so the benefit of membership is accessed to these specs. I mean, what else are they going to offer them in return for all this money?

Leo: Yeah, right.

Steve: So it's like, okay. But the problem is then none of the rest of us who have a real need and are not enterprises able to drop tens of thousands of dollars on dues to these things, I mean, and you have to be a member of all of them. So it's annoying.

Okay. So ROCA. We're beginning to see some specifics of fallout from the decade-plus use of very badly, poorly generated public keys by this broken embedded library. And there was something that teased me that I just didn't run down, but there was some suggestion that a shortcut was made in the algorithm where they had some collection, it was described as a "structure of primes" that were built in. And it's like, okay, it's hard for me to understand how that couldn't be really bad, so it must be different than I'm thinking. I mean, it just...

Leo: You don't want to use fixed, existing, well-known numbers as part of your factoring. That would be foolish.

Steve: No, no. That would not survive well. So we've learned a few things. First of all, Infineon, of course, within the industry, is a well-known company. They're the guys that had certainly the best of intentions, but they produced this defective library which Gemalto, which is a major smartcard and security access OEM, was using. They have a card which they began selling in 2004 which is called the IDPrime .NET card. They won't disclose, because now they're embarrassed, how many they've shipped. But it's believed to be as many as hundreds of millions. So hundreds of millions of individual Gemalto smartcards, based on the broken Infineon library, are not secure.

Now, it's probably not a coincidence that this card was discontinued at the end of last month, just this last month, three weeks ago, four weeks ago, at the end of September. Okay, we're done. However, the problem is they're the manufacturer. There are still third-party distributors with inventory still selling them. In Ars Technica's reporting, they contacted a Gemalto representative who referred them to a company advisory which read, quote: "Our investigation has determined that end-of-sale IDPrime .NET products may be affected." And so, okay, right.

Leo: What a coincidence. What a shock.

Steve: Yeah, we're not selling those anymore.

Leo: Yeah, you don't want those.

Steve: Good thing you didn't ask us last week. But now we're not selling them anymore. Meanwhile, cryptography experts who are curious, I mean, now you can imagine in the wake of this revelation from Monday before last, the industry is trying to understand what this means in practical terms. So cryptography experts who are now aware of this have been poking around. And they've added that there is little doubt of Gemalto's cards being affected. The CEO of Enigma Bridge, that is a security company, said he had examined 11 of Gemalto's IDPrime .NET cards issued from 2008 through earlier this

year. All of them used an underlying public key that tested positive for the crippling weakness. So you can extrapolate that to hundreds of millions.

So essentially what we have is an incredibly widespread, difficult to remediate, vulnerability. And this is the danger imposed by the use of widespread and defective consumer crypto. The problem is everybody's got these. There's hundreds of millions of them in people's wallets or pockets. And how do you deal with that? I think it's going to probably be up to individual companies to decide how they want to handle this, what they want to do. I mean, if they've got Gemalto products, they probably need to say, okay, how do we fix this? We need safe keys. And these are embedded in inexpensive cards, so they're not flash reprogrammable. They're dumpster discardable instead.

Also, and this news was on our radar last week, but I didn't mention it, that the government of Estonia that we've spoken of from time to time through the years because they're on the leading edge of deploying state-of-the-art identity for their citizenry. Unfortunately, they're using an identity system based on the Infineon library; and 750,000 electronic IDs it has issued, that is, the government of Estonia has issued, are vulnerable, three quarters of a million. And researchers have uncovered evidence that the ID cards issued also by Slovakia and Spain may also be vulnerable.

So essentially we have this widespread, deep, broad, and old vulnerability. And hopefully this will come to the attention of those organizations who are using these products. To be responsible, Gemalto should be sending a notice. Unfortunately, if they work through reps, they don't know who all their customers are. There's a layer of insulation between them. So, for example, unlike, I don't even know, I was going to use Yubico as a direct seller. I know that they know many of their end users. I don't know, though, if they run through a second tier of distribution. Oh, yeah, because you can buy them off Amazon. So they don't know who all their customers are.

Leo: Well, but they might be fulfilled by Yubico. Just because you buy it off Amazon...

Steve: Ah, good point.

Leo: My sense is they do, I don't know, that they know everybody. But that may not be correct.

Steve: So of course what we would hope is that any supplier, directly or indirectly, of technology that uses these soft public keys would take responsibility for that and say, you know, we profited from your trust, which, whoops, we didn't know we didn't deserve, so let's fix that. Let's hope that happens. The real threat isn't broad because, unlike something like Heartbleed, where a huge number of servers on the Internet theoretically had a low probability of being attacked, but when it happened the server could lose its private key, and then it could compromise it a lot.

Here, each individual card would have had a different public key. Each one could have, for on the order of $10,000 or so, have that public key cracked to determine the private key, which would then allow that card to be impersonated. So this suggests a huge vulnerability for targeted attacks, not broad and sweeping. But, for example, you could imagine any state actor who knows that a target is using, indirectly or directly, this Infineon library could be rubbing their hands together because for the first time, if they

can capture the public key, which the card readily gives you, so you just...

Leo: That's the point of the card.

Steve: Yes, exactly. It's like, here's my public key. And so that's distributed. So you just capture that, you crack it down, and you can then impersonate the person because the idea is that it offers the public key, and then it responds to a challenge which can only be answered by the private key. And that stays in the card. So the idea is, if you get the public key, which it broadcasts, for all intents and purposes, and crack it in order to extract the private key, now you can impersonate the cardholder for anything that they use the card for.

Oh, and that was the other thing, too. In some of the research I was doing for this, one of their customers - it was on the Gemalto site. It was one of their success stories. Some random wind farm company somewhere had equipped all 4,000 of their employees, and they were so proud, and they're emitting press releases because this one card does it all. And I'm thinking, it sure does. Now you only have to crack the public key once for a given card, and you can do everything.

Leo: Is it the same public key on all the cards?

Steve: No, no. Different public key. And so that's why it has to be targeted. You'd have to decide, okay, we want to get Willard. We want to be able to do what Willard can do, pretend to be him.

Leo: So you'd need Willard's card.

Steve: Yes, yes.

Leo: Having gotten his card, you get his public key. You defactor it. You know his private key. Give him the card back, and you can be Willard.

Steve: Exactly like any of the spy movies where you get Willard a little tipsy, and then you slip his card out of his wallet, run it through your reader, put it back in. And he still has his card, but you've got what you need. And then you send that up to a cracking farm, and in a short time you've got the ability to, without copying the card - because that's the whole point. The physical card is meant to be your security. And so the idea is there's only one of those that has Willard in it, essentially. And as long as he physically is in possession, it's the physical possession that is his security.

This breaks that by being able to leverage something the card freely gives out, the public key, being able to leverage that into something it deliberately never gives out, which is the private key. And notice, this is exactly the same model with the Trusted Platform Module 1.2, which many people have. I think it was 11 out of 41 laptops or something that we talked about last week. So not all, but many have exactly the same vulnerability. The TPM will say, "Here's my public key, woohoo," and anybody who asks for it can get it, the idea being that it'll never give up the private key. Well, now you don't need it to.

Leo: That's why you need two-factor.

Steve: Okay. In the interest - oh, I forgot to start my podcast timer, so we're about 30 minutes in. Normally I have that in mind. But okay. So I think we started around 2:00 o'clock. So, okay. This is the DUHK attack, but spelled D-U-H-K, which is an acronym for Don't Use Hard-coded Keys. Yeah, don't. Anyway, so I was tempted to say it should be the DUH attack, just D-U-H. But DUHK works better, and you need the "K" on the end anyway.

So two researchers at the University of Pennsylvania who Matthew Green often works with, and Johns Hopkins University's prolific Matthew Green, uncovered a significant implementation error in at least 12 commercial VPN and enterprise big-iron products. Fortinet is one that they highlight, only because - and it's just one of the 12 that they found. But Fortinet is a major supplier of industrial-grade, serious big-iron hardware. So I'm going to read the abstract from the paper. The link to the paper is in the show notes. And it is wonderful. I mean, the paper is really nice, for anyone who wants to really dig in and understand this.

So the abstract says: "The ANSI X9.17 random number generator" - so this is an ANSI spec, an official random number generator - "is a pseudorandom number generator design based on a block cipher and updated using the current time," that is, current time of day. "First standardized," they write, "in 1985" - so it's old - "variants of this PRNG [pseudorandom number generator] design were incorporated into numerous cryptographic standards over the following three decades. It remained on the list of FIPS 140-1 and 140-2" - that's the federal information something or other, oh, Federal Information Processing Standard. And the FIPS 140-1 and 140-2 are sort of the formal, these are what we approve of that anyone can use. And if you do, then you can stamp "FIPS certified" or "FIPS compliant" on your device. And so everybody does who thinks, oh, good, we want to be able to say that.

So it remained on the list until January of 2016, so about a year and three quarters ago they finally said, okay, this is old. Time to remove it. But as always, I mean, if we learned a lesson, any kind of lesson here, it's that, yes, you can say "stop using it," but you've got to make everyone stop using it. And no one has.

"The design," their abstract reads, "uses a static key" - and the static key is okay, that is, the design itself, the official design uses a static key - "with a specified block cipher to produce pseudorandom output. It has been known since at least 1998" - okay, so since this thing was 13 years old because it was born in '85 - "that the key must remain secret [yeah] in order for the random number generator to be secure." Actually, it was probably known in 1985, but it became obvious and apparent, probably.

"However, neither the FIPS 140-2 standardization process in 2001 or NIST's update of the algorithm in 2005 appear to have specified any process for key generation." That is, they just sort of say, oh, yeah, just a static key is fine, but they're not telling you how to make it. And of course all anyone had to do would have been to choose any source of entropy at all that the system is able to generate from packet arrival timing or anything to come up with a new key every time.

Anyway, they write: "We performed a systematic study of publicly available FIPS 140-2 certifications for hundreds of products that implemented the ANSI [that is, this thing], the ANSI X9.31 random number generator, and found 12 whose certification documents use static hard-coded keys in source code, leaving them vulnerable to an attacker who

can learn this key from the source code or binary." So what they did was they looked at the documentation specifically for devices and products that use these standards, suspecting that that might make them vulnerable, and then looked at what these guys were doing.

They said: "In order to demonstrate the practicality of this attack, we," they wrote, "developed a full passive decryption" - okay, listen - "full passive decryption attack against FortiGate VPN gateway products using FortiOS," which is their OS. "Private key recovery requires a few seconds of computation." Ouch. "We measured," they wrote, "the prevalence of this vulnerability on the visible Internet" - that is, just the public Internet, they were just out there looking at traffic - "using active scans and find that we are able to recover the random number generator state for 21% of HTTPS hosts serving a default Fortinet product certificate."

So Fortinet also terminates HTTPS connections, not just VPNs, but standard web connections; and, they write, "97% of hosts with metadata identifying FortiOS v4. We successfully demonstrate full private key recovery in the wild against a subset of these hosts that accept IPsec connections." So VPNs, HTTPS, and IPsec.

So, okay. What's going on? The problem is that this is old technology that has been deprecated. And this is a pseudorandom number generator whose use would be safe if its output was never visible. For example, if you used it to generate candidate primes for primality testing, and remember that a prime in such a situation is going to - one of them is going to be your private key. So it's by definition private. And if you got another prime, and you multiplied them, now you can't pull them apart practically, so both primes are now essentially secret, even though their product you can make public. The point being that that would be a fine use of this. Wouldn't matter if you had a hard-coded seed for this pseudorandom number generator because its output was never being broadcast.

What these instances - and I don't mean to just single out Fortinet, and these guys don't either because that was one of 12 different companies whose products were doing this - is one of the things that we've talked about, in fact during our discussion last week of the KRACK attack, part of the WiFi handshake is nonces being sent back and forth. Well, the nonce, and this is part of what the Diffie-Hellman handshake magic allows, is that somebody can observe each side sending nonces back and forth, and even with full knowledge of each end's - the nonce is a one-time random token. Even in full visibility, seeing these random tokens going back and forth, that doesn't allow a passive observer to determine the secret key that they end up generating, that each side generates, after receiving each other's nonce, which is really cool. But if these nonces are coming from this pseudorandom number generator whose output is never supposed to be seen, whoops.

So this is a misapplication of a pseudorandom number generator. And so what Matt and the two other researchers understood and figured out and then reduced to practice was capturing plaintext nonces which are being sent at the beginning of these various protocols as part of the key negotiation phase and the handshaking, grabbing those nonces, knowing that they come from this pseudorandom number generator whose output is never supposed to be seen, but that's what a nonce is. It's a public disclosure of randomness. And from just by capturing those, they're able to reverse-engineer the PRNG state, which is to say any secret information, like the key, essentially, that they may or may not know. And in this case the key is available to you, that is, you can reverse-engineer the firmware.

Sometimes the source code is published in some of these specifications, assuming that it doesn't matter. But if you know the key, and you capture a bit of its output, you can

figure out any unseen state in RAM that the device has, which allows you to both go back in time to recover previous state, which is to say previous nonces that would have been generated to decrypt conversations that happened before the conversation that you captured and the future. So you can go both directions in time because this thing generates semi-deterministic random numbers based upon a key which you then know.

So it's a beautiful piece of work. And the real lesson here is that it is very important not to have application drift. That is, 30 years ago, 20 years ago, 10 years ago people who were involved with this probably understood that this was very simple and convenient and, oh, look, it's FIPS-approved. Everybody thinks it's wonderful for this application. But you can't have application drift, where you start applying this to other things, without consequence.

So this has now been deprecated, but now it's going to have to be forcibly removed from the firmware wherever this is being used, thanks to Matthew and company's discovery that this not-supposed-to-be-seen pseudorandom number generator is not safe for generating public nonces. And our crypto protocols today use public nonces. You cannot have this generator generating them behind the scenes because we can determine the past and the future of the connections that it makes. It's very cool work.

So I need to engage you on this one because I think this is interesting sort of from a sociopolitical standpoint.

Leo: Oh, you're going to talk Kaspersky. I knew you were. All right.

Steve: Yes. Yup. Eugene says...

Leo: I have a strong opinion on this.

Steve: Yes. Eugene says, "We have nothing to hide." So they are planning, they have announced that they are going to open their software to external review and verification. So to give our audience, I mean, I know everyone knows about Kaspersky. They've got - their software is installed on 400 million computers worldwide.

Leo: Isn't that mindboggling, that number? Geez.

Steve: Yes. Yes. That number, it caught me by surprise. I thought, whoa.

Leo: That's got to be like 20% of all Windows computers. I mean...

Steve: 0.4 billion computers.

Leo: Amazing.

Steve: 0.4 billion. So in their announcement, Kaspersky did not name the outside

reviewers, but said they would have strong software security credentials - whoever they are, but we'll find out - and be able to conduct technical audits, source code reviews, and vulnerability assessments, as well as to examine Kaspersky's business practices and their software development methodology. And he said that they're going to open "transparency centers," because transparency is what they're saying they're trying to give us, in Asia, Europe, and the United States, where customers, governments, and others can access the results of these outside reviews and discuss any concerns about the security of their products.

And, finally, they're going to be expanding their bug bounty program from what was a paltry by today's standards maximum award of $5,000 to a more state-of-the-industry $100,000. Now, of course, in the reporting of this and kind of coming up to speed on this, I noted that not everyone's concerns were mollified. And I could argue that, I mean, I'm sure we could, that nothing Kaspersky could do would make any difference to those who just say, uh, we just don't trust them. So what do you think, Leo? I don't have an opinion either way.

Leo: Oh, I have a strong opinion. First of all, why, I mean, okay. Everybody deserves the benefit of the doubt, I guess. Eugene Kaspersky and his company have written software for the Russian secret police, the FSB. Because they're a Russian company, it's I would say likely that the FSB or the Russian government, or you know how they work through these shady third parties, could very easily compel them to include code in there.

Steve: And they could also have planted employees.

Leo: Absolutely. Why take the chance? I mean, already the Department of Homeland Security forbids, and the DoD forbids, any government agency from using Kaspersky. So if you absolutely - first of all, you and I agree we don't really believe in antiviruses. They just are another vector, frankly, into your system.

Steve: And in fact, yes, we've covered places where they introduce vulnerabilities that weren't already there.

Leo: Right. In many of the biggest named antivirus softwares. And, you know, our intelligence agencies are convinced that Kaspersky was instrumental in the exfiltration of NSA secrets from this contractor's computer. He brought them home, had Kaspersky on there.

Steve: Ah.

Leo: What they're not very clear about is necessarily where Kaspersky did it knowingly or merely provided a vector. Maybe there was a flaw in Kaspersky's software. But since it was the Russians that exfiltrated it, I'm just - I think that it's fine to give someone the benefit of the doubt if you're going to put them in jail. But I think the biggest problem is that Eugene Kaspersky, Kaspersky himself is beloved by American tech journalists. Dvorak loves Eugene and was always recommending

Kaspersky. I know a lot of tech journalists are reluctant to indict him because they know him.

Steve: Right, right.

Leo: I don't know him. I don't have any opinion on him.

Steve: Nor do I. And again, his company has grown large. He's no longer able, even if he's absolutely blemish-free, in a large company, I mean, it's like the executives of Sony weren't responsible for Sony's network being breached. It was a administrative assistant who clicked on a phishing link. And so you really can't hold the management of a company responsible for everything every employee in the company does.

Leo: Dave Redekop in our chatroom from Nerds On Site says, "Any sufficiently advanced antivirus software is indistinguishable from malware." That's somewhat true. But here's my - now, and then people will say, but yes, but he's opening the kimono. He's showing people his software. Here's my issue with that. That works in open source software because you can then download and compile the software yourself and assure that the binary you're using comes from the audited source code.

Steve: Right.

Leo: That's not what's happening here. He's giving source code to be audited, but there's no guarantee that's the source code that makes up the binary you're going to use; right? Or maybe I'm wrong.

Steve: No, no. I agree with you. And in fact it turns out it's surprisingly difficult for even two different developers to compile the same source and get identical binaries. They're just - it's like, if you use different versions of some libraries, you won't get a binary match. I mean, and if you hash it, you get something completely different. So it turns out it's extremely difficult to actually, even in the best of cases, to get binary identical output from separately compiled code because there are so many dependencies now, and every single thing has to be identical.

Leo: You modify one library, you give them the source code calling a clib, but then you modify a clib so that when it's shipped it has an extra line. It just phones home. Anyway, there's lots. But the thing that bothers me is their assertion that doing this will reassure people.

Steve: Right.

Leo: That makes me think they're up to no good. I mean...

Steve: Well, okay. My take is...

Leo: Maybe it's the best they could offer, I guess. I don't know.

Steve: Well, it's all they can do. But Leo, think of the economic impact.

Leo: Oh, yeah. I agree.

Steve: So this is devastating to Kaspersky that this is happening.

Leo: So that would argue, well, maybe we should give them the benefit of the doubt. But I...

Steve: Well, no. It just suggests that they're doing everything they can to hold onto market share because they've got a lot right now.

Leo: Well, when this was all brewing, they started giving it away, you know. Which made me even more - I was telling my radio audience listeners, I've been telling them for months, don't install any antivirus, but especially Kaspersky. Especially Kaspersky.

Steve: Well, stepping back from this a bit, there's sort of a broader thing, too, because I've noted on this podcast for years how surprised I am that Microsoft is able to sell Windows into China and Russia. I mean, when we hear they're using Windows, I always think, what? Really? They trust U.S. software? I mean, there's a contentious relationship here. And so, yeah, Kaspersky is selling their AV into us, and they're using Windows. And it's like, wow, okay. I mean, so it goes both ways. And I think it's surprising to me that in this day and age of there being open source alternatives to a proprietary OS like Windows, who in their right mind would use a black box where the hard drive is thrashing around, and it's constantly connecting, making random connections on the Internet. It's like, what's this doing? No. So, yeah.

Leo: Yeah, yeah. So I guess we're in agreement. And I feel bad. I mean, poor Eugene. I mean, maybe - I'm sure he's made enough money now that he can retire to Belize if he wants to.

Steve: He can just go find a beach. He'll find a beach. Yes, yeah, go to Belize and join John McAfee.

Leo: Kaspersky was good because it was one of the last antiviruses that didn't do a lot of heuristics, as I remember. It was all signature, so it was very fast and light. But nowadays I don't know how effective that is, even. Anyway, good.

Steve: Yes. So Google for Android, on their Android platform, is apparently, although we don't have an official announcement yet, going to be adding DNS over TLS, which is very interesting. Four days ago, on Friday the 20th, some guys over at XDA Developers - which as probably a lot of people know is a huge mobile platform developer community. I think it's something like 6.6 million members, and it's very active. One of them noticed something interesting in the Android Open Source Project (AOSP) software commits, where some changes were committed into the source.

And so they wrote, over at XDA Developers, they wrote: "It appears that DNS over TLS support is being added to Android, according to several commits added to the Android Open Source Project. The addition in the Android repository shows that a new setting will be added under Developer Options, allowing users to turn off or on DNS over TLS. Presumably, if such an option is being added to developer options, then it means it is in testing, and they arrive in a future version of Android such as v8.1."

Okay. So what does this mean? As we've often said, DNS, the Domain Name System, was never designed for security or privacy. In fact, you could argue it's probably one of the least secure, least private, but also simplest and thus fastest and leanest technologies, protocols, systems that we have on the 'Net. It uses UDP packets normally, which is also the lowest common denominator. It's just a payload on top of an IP packet which you just send somewhere. There's no setup. There's no handshaking. There's no, let's each of us agree on the numbering of the bytes that we're going to then be sending to each other in a continuous stream, none of that. That's all TCP, which is built on these datagrams.

UDP just says, I'm shooting this off toward you. If I hear back from you, I'll know that you got it. And if I don't, well, either you are busy, or it didn't get to you, or your reply didn't get to me, so we'll just try again. So, I mean, it just couldn't be any simpler. There's no encryption. There's no security. There's nothing preventing somebody from intercepting it and lying to you, bouncing it back sooner. In fact, we've talked over time about various ways of spoofing DNS, one being that, if you're closer to the person you're querying, and you see the query go by, and you inject a reply, if your reply arrives before the real one, yours is the one that gets accepted. I mean, it just couldn't be less secure.

So the problem was, as we were saying earlier, this was, from day one of the Internet, the DNS we have today is the DNS they designed decades ago. It was never designed for privacy and security. But in today's era of not only surveillance, but extreme spoofing and phishing, DNS's startlingly absent security and privacy is an increasing problem. And people are concerned about ISPs spying on them, seeing what they're doing. And even if you do encrypt your TCP connections under TLS, DNS, unless you go to other measures like using DNSCurve or DNSCrypt, which you have to take some measures to do, you don't have any privacy.

So DNS can run over TCP. By default it uses UDP packets on port 53. That's the well-known port for DNS. You can set up a TCP connection, though you don't have to. There are a few DNS management operations which must go over TCP because they transfer much more information. It's not just a simple query and reply. You're transferring whole zones, as they're called, over DNS. So there you need to do a zone transfer. You need to use TCP.

Okay. So moving forward, we know that DNSSEC, which we've been discussing recently, prevents forgery by having the DNS server provide signatures along with its responses so that the recipient is able to generate a hash of the responses and verify that the signature the server also provided matches, and then you use the public key which the

server publishes in order to verify that it had to have the private key to make the signatures correct. So, however, that doesn't give you any privacy. DNSSEC, let me say that again, DNSSEC, which hasn't even arrived yet, doesn't really work yet universally. It's coming, but there's no privacy. The replies are still in the clear. You verify their signature, but they're not hidden. So when we finally get DNSSEC, it's still no privacy.

Okay. So back last May, what is that, about a year and a half ago of 2016, May of 2016, RFC - and here again Request For Comment from the IETF, publicly, beautifully, bunch of people all worked on this in plain sight. They finalized RFC 7858, which is titled "Specification for DNS over Transport Layer Security (TLS)." Unlike regular DNS, which as I said uses well-known port 53 for UDP and TCP, TLS DNS uses 853, so port 853. They wanted to keep the 53. They said, hey, nobody's using 853. That one's ours. So that's what it will be.

Now, TLS does require some handshaking. So normally what happens where TLS is carried over TCP, you establish a TCP connection and then bring up a TLS tunnel, a secure tunnel, within that TCP connection, which gives you both authentication, so you know who you're talking to, and privacy, that is to say encryption. So you authenticate the endpoint, and you encrypt. So TLS, the use of TLS, even for DNS, still requires handshaking. So this is not as lightweight as DNS, but it makes sense for a number of reasons. For one, if there's no background bandwidth being drawn.

For example, Leo, if you and I were to go completely quiet and not move, there would still be substantial bandwidth back and forth between us because of the protocol that we're using. It doesn't go to zero. There's a constant background flux. Not so for TCP, if you're not - and we talked about this years ago when we were talking about the way TCP works. Basically, if neither endpoint is saying anything, there is no traffic at all. Essentially, by setting up this transaction, you establish some agreed-upon numbering for the bytes that you will be sending. And you can send a few, and the other guy can send you a few. But as long as you're both quiet, no overhead. Just there is some state, some memory that is committed at each end for maintaining an awareness of this agreement that you both have a connection. But there's no overhead. So that works in terms of does this make sense.

The other thing is that DNS tends to be bursty. That is, when you go to a new page, especially in this day and age, there's crap coming from all directions of the Internet in order to fill that page with ads and discuss content and just all the stuff. And so if those IP addresses for those domains, which may be a surprise to the browser, if they're also a surprise to your OS, then there's a burst of queries that goes out to your DNS server. And that's the other thing that works with the notion of establishing a relationship with a DNS server, which you otherwise don't in normal DNS. You just send off a UDP packet, and you hopefully get a response. But if you have established a relationship by bringing up a TLS tunnel, then you're always sending your queries to the same server, which generally systems are doing anyway.

So what all this means is that we bring up a flavor of TLS. This is not full HTTPS-style TLS with a certificate which is signed by a CA, which is trusted by the browser. There are two modes that this can work in because they're trying to keep this from getting too heavy-duty. They're just wanting to provide what they call "opportunistic privacy," where if you don't really - if you're not really worried about who you're connecting to, you'll still get a TLS tunnel for privacy. You may not know for sure who the tunnel is connected to, but at least somebody passively observing the traffic can't see the queries that you're making, any of them.

The next step is to use key pinned privacy, that is, so that this TLS DNS server running

on port 853, it'll hand you a certificate. And but it's not signed by a CA, it's just a self-signed cert, just asserting, here's a certificate, the idea being that there's some out-of-band means for you to obtain the pinning for that certificate's key. So the point is that maybe your ISP provided it. Maybe you established a full-strength HTTPS trusted and authenticated connection to obtain the pinning for that key once. Then you hold onto it, and then you just verify it. So it's very quick, very lightweight. Essentially you're verifying the signature of the certificate to verify that it matches the one that you have previously received because there's no way that we know of for somebody else to create their own spoofed certificate with a matching signature. That's cryptographically infeasible.

So what this suggests is that Google is continuing to move forward to, sort of in this chicken-and-egg fashion, to increase the privacy of Android and in general of the ecosystem. Now, this requires DNS servers that don't exist today. I mean, this protocol just got standardized last May or, yeah, May before last, May 2016. But the good news is it only uses already existing chunks of code. So it would be very simple for the various major DNS servers to bring up protocol support for this, like in a few days probably, and then test it and make sure they haven't created new vulnerabilities.

And then I would imagine that a major version or two, or maybe even in the next major version, we'll say, yeah, now we support TLS DNS, or DNS over TLS. And so that server would be opening a listening port on 853. Of course, firewalls would need to open, too, in order to allow traffic to come through. So there's some chicken-and-egg problem, but Google has, well, laid this egg.

And so we'll see what kind of uptake this has. But this is a good thing. It's lightweight, makes sense, and it looks like an Android version coming soon will at least, if it's on, probably attempt to bring up a connection to its DNS server on 853, which nobody else is bothering with yet. And when those answers start, when those attempted connections begin succeeding, we'll start having - Android users will start having some privacy, at least. An active man in the middle, because this is weakly authenticated, an active man in the middle could get in there; except that, if the Android user had previously obtained a key from that server, it could expect them to be the same. So a transient man in the middle wouldn't be able to break in and spoof.

So it's not uber strong, but it's really good. And it's lightweight. And I salute Google for saying, yes. We have a spec. It's a year and a half old. Let's add it to our client. And I wouldn't be surprised if we see other clients like iOS in the near future saying, yes, us, too. If Android's going to have it, we want iOS to have it. And then some servers will start supporting it. And before long, it'll be available.

And this is the way these things happen. Probably not quickly, but inevitably. And it seems like a good thing to do. It's a lightweight, nice compromise for adding privacy to our DNS. And again, ISPs, unless you VPN, they'll still know the IP address you're going to because that's still going to pass by. But it's another layer of privacy. And again, easy to do if it's just built into the infrastructure, as it will be before long.

Okay. So we're beginning to see a trend emerge which is not good. The name I couldn't think of at the top of the podcast was Elmedia.

Leo: I don't know this, and I don't know anybody who uses it.

Steve: Interesting.

Leo: So that's the good news. Well, everybody uses iTunes on the Mac; right?

Steve: Right, right, right, right.

Leo: And then there's [crosstalk].

Steve: Although I tell you, Leo, it's a nice-looking player. I took a look at it. It's clean, supports lots of standards. So what we're beginning to see is the emergence of what is being called "supply chain attacks," where something about the supply chain got compromised that caused malicious alteration of something, equipment, or in this case software downloads. Remember that in early May a download mirror of Handbrake, the very popular media converter, was compromised to download a Mac malware known as Proton, which we'll talk about in a second because it's coming back again. And of course a few weeks ago we talked about CCleaner. Same thing, one of their servers got compromised, and one particular version of CCleaner was altered maliciously.

And now last Thursday researchers at ESET discovered that Eltima, E-L-T-I-M-A is the company, Eltima's Elmedia Player - oh, and also they have a download manager. I have it in my notes here. I don't see it in front of me. Oh, there it is, Folx, F-O-L-X.

Leo: I'm not familiar with that, either.

Steve: Right. That one is even a little bit further off brand. But their media player, similarly compromised. And it was delivering Proton. Proton is a so-called RAT, a Remote Access Trojan. And it's pretty nasty. It first appeared last year as malware for the Mac and includes a range of features. It's able to execute console commands, access the user's webcam, log keystrokes, capture screenshots, open SSH/VNC remote connections, able to inject malicious code into the user's browser to display pop-ups asking its victims for information such as their credit card numbers, their login credentials, and other things, or like anything else if it's able to do a code injection into the browser. It's able also to hack the Mac user's iCloud account, even if two-factor authentication is in use. And in March of this year, so about six months ago, offered for sale on a number of cybercrime forums for $50,000. So it is potent sort of gray market, well, or black market cybercrime malware.

To their credit, Eltima was extremely responsive to ESET's report, immediately fixed the problem, notified their customers. If anyone who's listening is a - and I don't have a range of dates. It was just recently, so like in the last week or two, but I don't know when the attack, when this malicious change occurred, so I can't say if you downloaded this Elmedia player between these dates, that was the trouble. But in the disclosure of this, Eltima's instructions explained how to scan for three things. Under the temp directory, look for updater.app on your macOS. Under the library directory, look for LaunchAgents and then com.Eltima.UpdaterAgent.plist. Also under library look for .rand, and also under .rand look for updateragent.app.

They say: "The presence of any of these files above is an indication that your system may have been infected by the trojanized Elmedia Player or Folx application which means your OS X/Proton is most likely running" in that machine. The bad news is - oh, here it is. "If you downloaded Elmedia Player or Folx on the 19th of October 2017, your system

is likely affected." So looks like it was found quickly and removed quickly. But if anyone's concerned, if you are in Elmedia player, I've got the link to their page and disclosure. I'm sure you can find it on their site also.

Leo: They also recommend a total reinstall of your operating system. Holy cow.

Steve: Yes, yes. That's the problem. This thing cannot be removed safely. You just have to save your files and reload your OS. It's the only way to get rid of it. It is nasty. Okay, Leo. Donateyourtab.to.

Leo: Okay.

Steve: Donateyourtab.to.

Leo: They can have my tab.

Steve: Very cool. This is a sample cryptocurrency miner, asks your permission before it mines, and very politely...

Leo: Immediately crashed my browser. Okay.

Steve: Whoops.

Leo: You know, I wonder - this is Chrome. I wonder if Chrome's saying, yeah, we're not going to do that.

Steve: That's interesting. Well, it doesn't mine until you give it permission to.

Leo: Yeah, it's crashed.

Steve: It ran for me on Firefox. And I did have a bunch of people responding that it was working for them. Do you have a uBlock Origin installed on Chrome?

Leo: Oh, yeah, yeah, let me turn that off, yeah.

Steve: Yup. uBlock Origin will block it.

Leo: Thank you, uBlock.

Steve: Yes, exactly.

Leo: Nope, it's still crashing. I don't know what's going on.

Steve: Well, so this page, donateyourtab, all lowercase, no spaces, donateyourtab.to, runs a cryptocurrency miner. You're able to choose whether you want to help Puerto Rico through the Hispanic Federation; fight breast cancer through the Breast Cancer Research Foundation; the Rohingya, I think I pronounced that right, which is the highly persecuted minority Myanmar; or help civil rights through the Southern Poverty Law Center. You chose which of those groups you want to donate to, and then there's a huge iOS-style slide switch on the right-hand side of the screen. You drag that over. It then pops up a confirmation that we're going to run cryptocurrency mining on your machine. You say yes, and then it starts.

Down on the bottom is a slider where you're able to decide how much of your machine you want to give it. And in order to see how much performance you have, you can slide all the way to max. And then in the far lower right corner it shows you your cryptocurrency hashes per second, that is, the rate at which your platform on this browser, when set to max, is able to mine this particular cryptocurrency.

So in their FAQ they ask themselves the question: "How does the mining work?" They answer: "We use a service called Coin Hive, which is an API that allows browser-based cryptocurrency mining. All of the hashes you compute in your browser go into their pool; and once we reach," they write, "a certain dollar amount, Coin Hive sends us the money. We then cut a check to each charity based on how much was generated for each." They write, "We make ZERO [all caps] dollars off this. Coin Hive takes 30%" - and I choked when I saw that number, it was like, what? - "of all proceeds for running the mining pool and providing the mining services." Then they also wrote: "We're exploring ways to make that number lower by using a different or homegrown solution. We also use money raised to cover fees to run this site (hosting, taxes, et cetera)."

And of course Coin Hive is alone at the moment, so there's no competition. So 30% is what they're taking. If this were to take off, or if major vendors were to experiment with using this as revenue generation for visitors, as we discussed at length last week, certainly, well, that would go to zero, although they wouldn't be donating to charity. They'd be keeping the money themselves, rather than making us look at advertising.

But anyway, this followed perfectly on our discussion of this last week. And I thought just it's a - for all of our listeners who are curious, donateyourtab.to lets you experiment and also see how good your machine, how good your browser. Mine only got up to about nine hashes per second, which apparently is what an older iPhone can do. So, yes, my machine does everything I want it to, but it doesn't do it quickly. I'm seeing numbers of 80 and 90 being reported in my Twitter feed. And anybody who's curious can check my Twitter feed and see how people are responding because it varies. But only people with cell phones are as slow as my Windows, my old creaky, cranky, creaky Windows XP machine.

But very cool to see an example of this running. And I don't see a downside to it. It will be interesting to see if this ends up becoming something, if there is enough money in asking people to mine for the company that you're visiting. That would be an interesting model.

Meanwhile, Google has responded to bug reports from several users whose Chrome was

performing unrequested crypto mining. It turns out that in the Chromium project they're looking at natively blocking in-browser cryptocurrency mining. In the Chromium project, one of the recent bug reports wrote: "If a site is using more than XX%..."

Leo: That's what I'm using is Chromium. That might be it.

Steve: If a site, well, I don't think it's deployed yet because they're still working on it. And you would think they would give you a pop-up or do something that was a little more gentle. "If a site is using more than XX% CPU for more than YY seconds, then we put the page into 'battery saver mode' where we aggressively throttle tasks and show a" - well, their internal term is a "toast," which is a notification popup, they call them "toasts" - "allowing the user to opt-out of battery saver mode. When a battery saver mode tab is backgrounded, we stop running tasks entirely."

They write: "I think we'll want measurements to figure out what values to use for XX and YY, but we could start with really egregious things like 100% for 60 seconds." And finally he finishes, saying: "I'm effectively suggesting we add a permission here, but it would have unusual triggering conditions. It only triggers when the page is doing a likely bad thing." And then subsequently there was a lot of discussion back and forth. No consensus yet reached.

But I appreciate that Google is being proactive. And certainly, because of the battery drain problem, you certainly don't want mobile devices to be doing this without your knowledge and your permission. And we would like our desktops to advise us that this is going on, like maybe pop up "This page is saturating your processor. It's going to make other things sluggish. Would you like us to throttle it for you, or shall we proceed?" And "(It may be doing mining, which you're not getting any benefit from.)" Who knows.

And of course there are, until Google and if Google should add in-browser blocking of mining, there is an AntiMiner extension, there's one called No Coin, and there's minerBlock as explicit blockers. And then, as we mentioned last week, AdBlock Plus is adding it, or has now, and uBlock Origin has had it blocked for some time. And apparently some AV products can also block those miners. So anyway, even if you don't do this, having the ability to play with it I think is cool. And I know that people who picked up on this earlier today from my Twitter feed thought, hey, this is cool. And I began getting a flood of reports back about how many hashes per second people were doing.

Also on the Google track, they're adding what they're calling Advanced Protection to Chrome. Google has partnered with ESET to beef up their detection of inadvertently downloaded and also removal under Chrome's "Cleanup" feature. So Google explained, they said: "We upgraded the technology we use in Chrome Cleanup" - which is already a thing - "to detect and remove unwanted software."

They write: "We worked with IT security company ESET to combine their detection engine with Chrome's sandboxing technology. We now detect and remove more unwanted software than ever before, meaning more people can benefit from Chrome Cleanup. Note this new sandboxed engine is not a general purpose antivirus. It only removes software that doesn't comply with Chrome's unwanted software policy." But now enhanced with technology from ESET, which again I think is very cool.

Leo: That's not the Advanced Protection. I think that's in the same article, but I think that's just a separate thing because Advanced Protection is that thing we talked about last week.

Steve: Correct.

Leo: Do you want a report back on my experience?

Steve: Oh, yeah, yeah, yeah, yeah.

Leo: Because remember I ordered - I already had a YubiKey. These are unique...

Steve: Yes. And that's the little round - the little BTLE.

Leo: The VASCO DIGIPASS, yeah.

Steve: Yes.

Leo: So these are both FIDO UTF keys. YubiKey always was. I had a YubiKey 4. So I ordered, yeah, this is the Bluetooth one. You press the button, and you have to have a Google trusted application on your device that recognizes this and authorizes you, and then you can use Google apps. But there's some severe downsides to this. In fact, so much so I turned it off.

Steve: Yes.

Leo: So one is you can't use - and you mentioned this - third-party apps, which means none of your Apple iPad, your calendar, you have to use Google's apps entirely. You can't use Apple's calendar, Apple's email, any of the Apple stuff, or any other third-party stuff.

Steve: To access your Google property.

Leo: Which is as it should be. I mean, if you want to really lock it down, Google can only lock it down if it doesn't give permission to any other apps, any third-party apps.

Steve: And this is why it's always good to remind everybody this is not for everyone.

Leo: Yeah. Yeah, oh, boy, is it not. The other problem that was the deal breaker for me is, as far as I could tell, it doesn't work on Linux. So Chrome or Chrome. You do have to use Chrome as your browser. So I installed Chrome on Linux, and it still didn't work. Maybe a bug. I've seen this kind of authentication bug before with Google with their phone login techniques. So it might be a bug.

Steve: Yeah, that's going to be an OS-dependent driver thing, too. So you could see their...

Leo: I tried it on all my different Linux machines with different distros and so forth, and it just - the YubiKey works fine with other apps. But as soon as you put the YubiKey in, it bombs out. So I don't know if that's something they'll fix. They probably will. If you can live entirely in the Google ecosystem, this is a great idea. And it's not, I mean, it doesn't cost - to answer the question we weren't sure about last week, there's no additional cost, once you buy these keys.

Steve: Ah, right, right, right. So there's no, like, service subscription or anything from Google.

Leo: No, no fee.

Steve: Nice, nice.

Leo: And you don't have to buy the ones they recommend, which is a good thing because this is out of stock now, as well. But YubiKey and the VASCO DIGIPASS worked fine. You need one USB and one Bluetooth LE key. And you use one for mobile, any device that doesn't have a USB plug, and one for desktop. It works well. It's a really nice procedure. You only have to authenticate it once on each device, just like always; right? It puts a cookie on your system, says oh, no, he's him.

Steve: Ah, good. That's very good. So it's not a per-use reauthentication.

Leo: No. It's not burdensome in that regard. It's only the first time you use a new device with your Google account. The biggest burden...

Steve: And that makes sense, too, because the threat model is some third party somewhere else.

Leo: Right. Some guy saying, oh, I've got Leo's password. Now let me log in.

Steve: Yes, yes, yes.

Leo: I think that the only thing that makes this - the two things that make it different, one is there's no fallback. So if you lose these keys, you have to go through this stringent account reauthorization which takes time and requires humans.

Steve: And, unfortunately, that's as it should be.

Leo: Again, as it should be. And you can't use third-party apps. But really Google's two-factor you can narrow down to just - I turned off text messaging, for instance. Just the Authenticator and something else, including a YubiKey. So you can get many of the benefits just turning on two-factor and making it a little bit more locked down.

Steve: Right, right.

Leo: Anyway, I didn't mean to interrupt, but this opens [crosstalk].

Steve: No, no, this is perfect.

Leo: I thought I'd finish it.

Steve: Yeah. That's good. I'm glad you did. So just a quick update on our buddy Marcus Hutchins, aka MalwareTech. He may no longer be under curfew or GPS monitoring. Marcia [Hofmann], I don't remember her last name, one of his attorneys, argued successfully last Thursday that the restrictions that had been imposed on him were needlessly burdensome. He had never missed a court appearance. Even when his GPS device failed during a trip to the East Coast, he did not attempt to flee, she pointed out.

And he apparently wants to surf and swim since he's in Venice Beach in the L.A. area on the West Coast of California, where the weather is fine. And so this is needlessly, she's arguing, impeding him to have to wear a GPS tracker. So the judge agreed and released him from those obligations. And in fact he tweeted on Saturday, he said: "Californians claiming it's cold, meanwhile I'm wishing there was a way to wear less clothes than I'm wearing without being arrested." So it sounds like he's having fun in Southern California.

However, a day later, last Friday, the DoJ filed a motion to revoke the judge's decision. So they're just going to fight. Michael Chmelar, who is an Assistant United States Attorney said: "While it's true pretrial services possesses the defendant's passport, it is unrealistic to think that the defendant could not leave the U.S. without travel documents." So it's like, okay. And what is the GPS's - I guess that forms a geographic tether? So if he was at an airport, would alarms go off?

Leo: Oh, yeah, yeah.

Steve: Maybe TSA would wonder why there's...

Leo: It's probably one of those house-arrest bracelets; right? I don't know, but...

Steve: Yeah, well, it's GPS, so it's global. And he was apparently traveling across the country with it. So somebody wants to know where he is all the time.

Leo: Yeah, yeah.

Steve: Yeah. Anyway, he later tweeted, after the motion to revoke, that he was still in limbo. So he doesn't know what's going on, and nobody apparently does. So anyway, we'll see how this turns out. I think we're probably correct, Leo, in that there's some history that he's going to be held to account for, unfortunately.

Okay. I mentioned at the beginning of the show the Windows 10 Fall Creators Update adding something called TruePlay. And I thought, what? Anti-game cheating technology. And I kind of grumbled to myself, "Because Microsoft had some spare time and nothing else to do?" But apparently this is a thing, and I'm sure you probably know about it, Leo; and I'm sure Paul does. I guess that, for example, something, the online gaming, for example, Grand Theft Auto, there are services, subscription-based services that provide the ability of players to do all kinds of cheats of the game, providing endless cash, teleporting...

Leo: Huge problem, yeah.

Steve: Yes, teleporting them to arbitrary places, becoming invulnerable, walking through walls and so forth. So it turns out...

Leo: It's not so much a problem in single-player games.

Steve: Right, because you're just cheating yourself.

Leo: Yeah, I mean, those are God Mode. So somebody cheating in GTA, who cares? But on online games, that's a pain in the tuchus.

Steve: Right, right. So it turns out that developers who went to the effort could increase the isolation of their game, that is, their existing technologies in Windows for creating better process isolation to prevent most of this from happening. Microsoft has decided to incorporate that, essentially submerge that into the OS.

So there's a technology in Windows apps known as a manifest, which is a file that a developer can create - I have one for SQRL, for example - that declares some things to the OS about this application that it's getting ready to run. So it is bound into the executable. The whole thing is signed so that it can't be changed. And when Windows loads the EXE, the executable, it looks at the manifest to learn things like what level of UI it's familiar with, how much security it wants, and various things.

Well, a new item, a new entry in this manifest can be the declaration that it wants TruePlay enhanced isolation. And so Windows 10, I guess it's now in beta, but it'll be - I mean this feature TruePlay is in beta. It will be incorporated into the Fall Creators Update. It will recognize this manifest and automatically sequester this app from other cheats which would otherwise be able to get into it.

So from a security standpoint it's kind of cool because it makes it very simple for developers not to have to keep reinventing and reimplementing the same technology, but simply to say to Microsoft, "We would like this to be a protective process running on the platform." And the other cool thing is that then, as Microsoft finds failures in their protection, that is, as the attackers work around what Microsoft has done, Microsoft can evolve the protection to match that and, as a consequence, all of the protected gaming technologies, or anything that wants to run in the TruePlay sandbox, essentially, gets the advantage of that. So a nice advance from a security standpoint, too. And solving a problem for online game players.

And speaking of solving a problem, Rich Williams in Daphne, Alabama, just yesterday on the 23rd, sent a note titled "One More SpinRite," and he meant success. He said: "Hi, Steve. I know you receive a lot of feedback and have discussed SpinRite and its ability to repair many storage devices. But I ran into something last week you and listeners may want to know." And actually we have discussed this a couple times just coincidentally with Father Robert when he was on the show because the idea of SpinRite fixing smartphones happened to come up, and Father Robert had actually done that on one of his.

He said: "My Samsung S7, which runs Android, has been having a problem recently where I would regularly be notified that my SD card was encrypted." It turns out, he says: "This is normal at boot, and everything seemed to run fine, so I didn't worry about it at first. But after some research to stop the annoyance, I found that it was being caused by an unmount/remount operation happening when Android failed to read the card." So there was beginning to be some flakiness, he's saying, essentially, in the card that was causing Android to react badly to a read problem.

He said: "I store my podcasts, pictures, and videos on it, and it holds 250GB so I can't afford to start losing data gradually. Needless to say," he wrote, "I pulled out my copy of SpinRite. And after it completed running on Level 2" - which is what you want to run on solid-state media - "with no errors, it's been three days now and no more alerts. Thank you for a great product and for the work you do on Security Now!."

So I would call that a refresh operation. Much as we used to do on hard disks to just refresh the low-level format, this running SpinRite on an SD card that's beginning to get flaky refreshes the data, strengthens it, and actually we're not writing, remember. Level 2 is just a read. But the card's controller is watching. And SpinRite does flip some switches that say we're doing maintenance things now, so pay attention here. And when the SD controller sees that the card had trouble, it says, ooh, and will then do a rewrite only of those regions which need repair.

And that's why SpinRite says everything was fine because this was all down a level below the data access where the actual data, where the bits are being stored and the controller interacts and just reports, okay, everything was fine. And then behind the scenes it makes sure that stays that way by rewriting the troubled data. So another example of SpinRite's future on solid-state drives, although it sure doesn't look like spinning drives are going away anytime soon. They just keep doing - the cost per bit just keeps falling, or per byte, and with no end in sight.

Looking at our time, I'm going to skip our feedback because we're coming up on two hours here, and I want to talk about Flash IoT Bots. So, okay. As I said at the top of the show, Mirai - this is literally, like, a few days ago, the one-year anniversary of the discovery of Mirai. And as we know, it was about 100,000 devices that were sufficient to distributed denial of service, DDoS attack the DynDNS service. That botnet pushed DynDNS off the 'Net and held it off long enough for the other DNS servers who had made a request some time ago for their caches of those requests to expire. They went back to try to refresh the cache with updated IP address information, if any, and DynDNS was off.

So after a while those second-tier DNS servers removed the expired DNS data from the cache. And now anyone asking that DNS server for the IP address of a domain that it had to go to ask DynDNS for got a "Sorry, we don't know that domain." And so major sites, surprisingly major sites, were pushed off the 'Net by this attack on one DNS provider. So that was with 5% of the number of bots that are now under control of a botnet that was discovered a week ago. Because it was independently found by two different security firms, they each gave it their own name. One called it the "IoT Reaper" because it is reaping IoT devices. The other called it, having some fun with the IoT initials, called it "IoTroop" because it is that, too, a troop of IoT devices, although it's not...

Leo: I like Reaper. I think that's best. Reaper's good.

Steve: The Reaper, yes. So as we'll recall when we discussed this a year ago, Mirai - god, I can't believe it's been a year, Leo. It seems like it was just the other day.

Leo: I know.

Steve: Like where did a year go? Mirai was doing telnet scans, scanning the 'Net, port 21? 23? I've forgotten what telnet is.

Leo: Is it telnet? Yeah, 21.

Steve: Yeah, 21. Scanning the 'Net for responses on 21, and then it had a list of default or weak admin WAN logon, admin credential logons. And it would just guess usernames and passwords. And for a frightening number of cameras and DVRs that were exposed on the Internet with their telnet port flapping in the breeze, the Mirai botnet was able to get in.

Now, a botnet which itself is a scanner is frightening because that's where the scale comes from. That is, one gets launched. It starts looking around for others. And remember the old days of the worms on the Internet, where they just, like Code Red and Nimda and MSBlast, these things explode onto the Internet because each one that it gets infected starts looking for others to infect. And so you get this exponential increase. Now, that's also been recognized in the past as...

Leo: Port 23, sorry. FTP is 21. I confuse my canonical ports.

Steve: Oh, 22. That's right, FTP is 21.

Leo: Telnet's 23. SSH is 22.

Steve: 23, there we go.

Leo: It's so confusing, 23.

Steve: That's what I thought, 23.

Leo: You were right.

Steve: So you get this exponential growth. Now, this botnet is deliberately - it recognizes - the designer, the author recognized that there were going to be a lot of these, and so the scanning per is kept on the DL. These things do not aggressively scan, probably to keep people from thinking, what the hell is going on with my bandwidth, if you had one of these that were infected.

Okay. So here's the difference. Mirai was looking for open telnet, guessing usernames and passwords using simple ones or default ones. This nasty bot is using known weaknesses in existing products, many weaknesses known for years. And they're in D-Link, Netgear, Linksys routers; GoAhead, JAWS, and AVTECH cameras; and a Vacron Network Video Recorder. And this author is staying abreast. In fact, in the show notes, second from the last page, Leo, I have a chart showing the nine vulnerabilities that are being used across these various devices. However, Check Point also spotted the botnet attacking the MikroTik and the TP-Link routers, Synology NAS devices, and Linux servers.

Leo: What?

Steve: So this thing is a - yes, it is aggressive. And number five there on that chart, you can see where this Vacron NVR Remote Command Execution vulnerability was discovered on October 8th, and it appeared in the wild in use two days later.

Leo: Oh, wow.

Steve: In this botnet. Meaning that the author is watching the industry actively and immediately incorporating anything that is known. And what's distressing, for example, is that the multiple vulnerabilities - number seven, multiple vulnerabilities in Linksys E1500/E2500.

Leo: Look when that was discovered.

Steve: Yes. February of - oh, no, I'm sorry. Yes, in 2013.

Leo: 2013.

Steve: 2013. So more than four years old, and there are still Netgear. These are...

Leo: D-Link, too.

Steve: Yes, yes. And Netgear DGN devices, D-Link, yes. And so here's the problem. These are IoT. They're appliances. They're on the 'Net. They're Internet-facing. I mean, D-Link, Netgear, and Linksys, those are routers. They're designed to be on the 'Net. They've got longstanding known vulnerabilities. Nobody ever patched them. Now this botnet is finding them all and installing itself in them. And we are now, we're at two million compromised devices, and it is growing at 10,000 devices per day. It is going to find them all.

And there's a full Lua environment built into this thing, the full Lua execution environment, suggesting that the author intends to and can write very complex and efficient attack scripts. And there are more than 100 Open DNS IPs in the firmware, meaning an Open DNS server, "open" meaning it will accept a query from anyone, rather than only like the customers of its own ISPs. And of course that's what you want for DNS reflection attacks, where you spoof the IP of a query that will generate a much larger response.

And the unwitting DNS server will send the response to your attack target so that - can you imagine if more than two million of these well-connected routers on the Internet start sending queries off to someone that this person, just because they can, wanted to push off the Internet? I can't, I mean, the bad news is we will be discussing in a week or two or three or whenever, because it's clear this is not just a research project. If you've got 100 Open DNS server IPs built into this thing, the only thing they're good for is reflection attacks.

So we will, without question - a year ago it was DynDNS. We don't know what this person has in plan. But this is a very serious IoT net. And I wanted to step back a little bit and just note that, you know, what are the characteristics here? We've got devices with longstanding, in some cases more than four-year known vulnerabilities. But people don't think of them as a computer. They think of it as the box in the closet, the box on the shelf, the little piece of plastic they bought down at the electronics store for $49. They hook it in and, oh, it seems to be working just fine.

Yes, and unfortunately there were problems that were found, and this thing isn't updating itself automatically, unlike the Ring products that you were just talking about. Instead, you've got to go do something. Nobody ever will. So this is now an IoT device, sitting on the Internet, which is going to be found by this exponentially growing IoT Reaper flash botnet. And how do you fix this? How do you remediate this problem? I just don't think you do.

And, I mean, we knew about these problems. When I was doing the research, I ran across a posting on the Hacker News in early April 2014; right? Early April 2014. So just over 3.5 years ago. It was titled "Millions of Vulnerable Routers Aiding Massive DNS Amplification DDoS Attacks." Exactly this, 3.5 years ago. Nothing happened. So now something's going to happen, and we don't know what.

Leo: Whew. Wow.

Steve: Yeah. So we have a problem because we have an industry where an install base of non-self-updating devices which are vulnerable are sitting on the Internet and are going to become attack platforms for the future. And I coined the term years ago, IBR, Internet Background Radiation, because there's still Code Red and Nimda packets out there from old Windows machines in the closet that are still out there hoping to get lucky. They're not going to very often, but they're going to a little bit. And, boy. I mean, this is an ongoing problem. I don't know how we solve this. But the good - I was going to say the good news. The bad news is it's going to take a catastrophe, and this is a potential catastrophe.

Leo: Do you think a nation-state actor or individual?

Steve: No, there's been no claim. I haven't read any supposition. The IPs that are being used are known. The botnet is generating candidates for infection and sending them to a command-and-control server.

Leo: You could bring down the C&C server; right?

Steve: Except that new ones are being added constantly in order to scale. I mean, just think about it. Two million devices are phoning home to somebody. You need a network of command-and-control just to ride herd over this herd.

Leo: Yeah, yeah.

Steve: It's amazing.

Leo: Wow.

Steve: You need a botnet just to control the botnet.

Leo: It's amazing.

Steve: Yeah.

Leo: And scary.

Steve: And, yes. If this thing went peer-to-peer, if they moved away from a command-and-control model to a peer-to-peer model - and I hope I just didn't give the guy any ideas, although I don't think they need any ideas from me. If they've put a Lua execution

environment in there, this thing can support some code to do anything they want. But if it switched into a peer-to-peer mode so that it became independent of command-and-control, then we've really got some problems.

Leo: How would you control such a thing?

Steve: All you would have to know would be your own private access. For example, you could send commands that were cryptographically signed by your private key. Nobody reverse-engineering it could figure out what your private key was. So nobody else could send their commands to your botnet in order to take it down. And so you need a secret. But once you have that secret, you could allow your bots to verify that commands are coming from an authority. You can't have them talking to each other that way because then they would have to have a key that could be spoofed. But the controller of this network could have a private key which is used to sign authenticated commands and then just spray them. Just spray the commands.

Leo: Or have them come and spread it automatically, peer-to-peer. Just say, hey, if you get this command, make sure you tell your - it's like a cell structure or something.

Steve: Tell everybody you know, yes.

Leo: Yeah, yeah. Well, and that's of course in espionage what you use. You use cells so that even taking out a cell doesn't take down the whole thing. And no one cell knows more than two or three other participants.

Steve: Yup.

Leo: I don't know why my mind's working this way. I'm now thinking about how we would set this up, Steve. But don't worry.

Steve: Any more good ideas, Leo?

Leo: Don't worry. I won't be - oh, as you say, I doubt very much we're telling people anything they don't already know.

Steve: No, no. This guy...

Leo: I really feel like these are - at some point, something this large, this is for taking down the grid or something. I mean, the last time they used it to take down the PlayStation network.

Steve: Well, yes. As I said at the top of the show, what could this do? Anything it wants,

literally. Anything. Two million.

Leo: It could take down the whole Internet, I guess; right?

Steve: Two million bots doing whatever they want to do, yeah.

Leo: You can't - the 13 root servers for the DNS system, those would be hard to attack. I would imagine they're secured in a variety of ways.

Steve: Well, and there are only 13 logical, yes.

Leo: Yeah, they're multi-routed, multi-owned.

Steve: Actually they're widely distributed, yes. So it's only 13 IPs. There's actually a ton of actual physical hardware.

Leo: Right. So it'd be hard to take that down.

Steve: Yeah.

Leo: Well, if it's North Korea, maybe they just go after Sony again, and then we can all rest easy. Or it could be Elliot, and could be Stage 2 on E Corp, as somebody in the chatroom said.

Steve: I don't think we'll be waiting long to find out.

Leo: It's Evil Corp's botnet. Mr. Gibson, as always, you've scared us and amused us and reassured us all at the same time. And I think the scare will last a little longer than the rest of it. We will be back next week to do more of the same. You'll find Steve...

Steve: If the world still exists, if the Internet still exists, yes.

Leo: Wow. It's really intriguing to think about.

Steve: Yeah, I know, it's - yeah.

Leo: How big was the old one, the Mirai?

Steve: 100,000.

Leo: So this is...

Steve: Two million.

Leo: This is 20 times bigger.

Steve: Growing at 10% of the old one per day. They're adding 10,000 new devices per day.

Leo: Oh, interesting. Yeah, you don't build that for nothing.

Steve: No.

Leo: Okay. Well, thank you, Mr. G., for all your insights.

Steve: My pleasure. Always happy to shine a little sunshine on something.

Leo: You can find Steve at his website, GRC.com. That's where you'll find audio and transcripts of this fine show, all 634 episodes. You'll also find many of his fabulous programs, including of course SpinRite, the world's best hard drive maintenance and recovery utility. You can find out more about SQRL, soon to be birthed into the world. Also Perfect Paper Passwords and ShieldsUP! and DCOMbobulator and on and on and on. He's done a lot of great projects. His Healthy Sleep Formula. There's all sorts of stuff. It's fun. It's like Grandma's attic or something. You just want to browse around for a while.

Steve: And some of the things are that dusty.

Leo: They're that old, yeah. Like DCOMbobulator. I don't know what made me think of that. You can also find the show on our site, TWiT.tv/sn. In fact, we have a link or links there to subscribe to all the favorite podcatchers, YouTube and all of that. You can watch us live during the taping of the show. We have an open studio policy. Either come by and say hi, email tickets@twit.tv, or watch the live stream at TWiT.tv/live. You can say "Echo, listen to TWiT Live," and it'll play it for you.

You can also join us in the chatroom, if you do, because that's the best way to talk, the kids in the back of the class, always talking around. It's fun, and it's a great social network on its own of true geeks. That's irc.twit.tv. If you can't watch live, tweet live, chat live, you can always get on-demand audio and video after the fact at our website, TWiT.tv/sn or wherever you get your podcasts. Make sure you subscribe. This is one you want every episode of. And don't forget you can reach

Steve by tweeting him, @SGgrc. He accepts DMs of any length.

Steve: I do.

Leo: Thank you, Steve. And we will see you next time.

Steve: Okay, my friend. Back next week for 635.

Leo: Yeah, baby. Bye-bye.