SECURITY NOW!

Transcript of Episode #602

Let's Spoof!

Description: This week, Leo and I discuss the countdown to March's Patch Tuesday. What was behind Amazon's S3 outage? Why don't I have a cellular connectivity backup? We share some additional Cloudflare perspective. Amazon will fight another day over their Voice Assistant's privacy. An examination of the top nine Android password managers uncovers problems. We'll cover another fileless malware campaign found in the wild; security improvements in Chrome and Firefox; a proof of concept for BIOS ransomware; a how-to walk-through for return-oriented programming; a nifty new site-scanning service.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-602.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-602-lq.mp3

Matthew Green compares desktop and mobile security. We've got a bunch of feedback quickies, an incredibly wonderful waste-of-time accomplishment, the future threat of deliberately fooling AI, and the dark side of automated domain validation certificate issuance.

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. He's got all the security news. He's got a lot of other stuff, too, including the weirdest scientific diversion ever, and a look at some new certs from Let's Encrypt that could be problematic. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 602, recorded Tuesday, March 7, 2017: Let's Spoof!

It's time for Security Now!, the show where we get together and talk about security with the one who knows, Mr. Steven "Tiberius" Gibson of GRC. Hi, Steve.

Steve Gibson: Yo, Leo. Great to be with you. This is the first Tuesday of March because March 1st was last Wednesday. So this is, once again, this is the lead-up to the probably big March second Tuesday, which of course - so I guess this would be the, what, the penultimate Tuesday...

Leo: Use your words.

Steve: ...before the March Patch Tuesday, which we expect to be huge, well, huge in the sense of it fixing a lot of things. But as we know, now that Windows has just a single blob that is a rollup of everything that they fixed, it's just going to be one thing, but it ought to be a big relief for people who are concerned. I've even seen some independent companies releasing short-term fixes, their own Windows patches for some of these things that Microsoft just didn't get around to fixing in time.

And we've got a ton of news. This is Episode 602. We now know what was behind Amazon's S3 outage. I wanted to just quickly respond to why I don't have cellular connectivity backups, backup connectivity, which a number of our listeners commented on due to last week's delayed recording. I wanted also to share some additional Cloudflare perspective, following up also on last week. Amazon turns out is going to be fighting another day over their Voice Assistant system's privacy questions, so we'll touch on that briefly. An examination of the top nine Android password managers, which uncovered problems in them all, including LastPass, the one that I use and you use and is sort of the show favorite.

There's another wireless malware campaign which has been found in the wild using a disturbing new technology - well, actually a disturbing old technology in a new way. We've got security improvements in both Chrome and Firefox that are both interesting in different ways. A proof-of-concept BIOS ransomware, which was demoed at the RSA conference and will be fully disclosed at this upcoming Black Hat Asia conference at the end of the month. A brief how-to walkthrough for return-oriented programming. I'm not going to go into it in depth, but it's interesting, and I know that some of our listeners will want the details. I found a nifty new site-scanning service. Matthew Green in his most recent blog compares the fundamental security of desktop versus mobile in a very interesting way that I wanted to share because I knew that would be of interest.

We've got a bunch of feedback quickies from our listeners. The most wonderful, amazing waste-of-time accomplishment I've ever seen. I mean, like, okay, I mean, it's just a feat that I think you and our listeners and I also really got a big kick out of. Then the title of this podcast is how we'll wrap up, after we talk about something that just hit me between the eyes, and that is the future threat of deliberately fooling AI. Then we'll wrap up with the reason I titled this "Let's Spoof!" - the dark side of automated domain validation certification issuance.

Leo: Oh.

Steve: That is, we know about Let's Encrypt.

Leo: Let's Encrypt, now Let's Spoof.

Steve: And what's happened, Let's Spoof!. Turns out it's being horribly abused.

Leo: Of course it is.

Steve: And, you know, of course it is.

Leo: This is why we can't have nice things.

Steve: Exactly.

Leo: You're not going to talk about the Vault 7 - it's probably too early to talk about the WikiLeaks Vault 7 release of CIA tools.

Steve: I just didn't have any time to come up to speed.

Leo: Yeah. And we'll talk, I'm sure, next week about it. And it's one of those things where it's 7,000 pages. It's a lot to look at, millions of lines of code. I do think it's interesting the names. You remember we were always fascinated by the naming scheme that the NSA had used in the Snowden dump.

Steve: Right.

Leo: But that was, I guess, a randomized two-word generator.

Steve: Right.

Leo: Apparently the CIA doesn't have such an interest - or the GCHQ because I think of these came from Britain - in obfuscating the purposes and the functions of its tools, though they have a lot of clever names that do in fact give you a hint as to what the thing does. I'm sure we'll be talking about that a lot over the next few weeks. All right.

Steve: I forgot to ask you how many breaks we had in this podcast.

Leo: One more after this.

Steve: Okay. So our Picture of the Week is xkcd having some fun with a topic we've been covering a lot. This is xkcd.com/1807. And the cartoon shows his little stick figure people that he typically uses, with a couple entering the house. The people who live there say, "Hello, welcome to our house." And then the couple entering say, "Thanks for inviting us," and then the A-word.

Leo: Echo, we'll say Echo.

Steve: Echo.

Leo: Yeah.

Steve: So they say, "Thanks for inviting us. Echo, order two tons of creamed corn. Echo, confirm purchase." And then the caption says: "When visiting a new house, it's good to check whether they have an always-on device transmitting your conversation somewhere."

Leo: Wow. Come on, Randall. Really? Aw.

Steve: And of course we know that's not the way it works.

Leo: Somebody made a point, though, the other day that you could, Amazon could program it to listen for other keywords. Right? I mean, sure, there's not a lot of processing or memory in that, but maybe a handful of other keywords that it would silently transmit up to the cloud.

Steve: Well, Leo, they could change...

Leo: They could just turn it on.

Steve: Yes. The problem is they can't have all of the devices in the world simultaneously streaming audio to them. That would probably even tax them.

Leo: Yeah. And nobody's got time for that.

Steve: But they could certainly turn on a microphone here and there, if they had some cause to.

Leo: I suspect that's why they fought the law enforcement subpoena, because they don't want the next step to be law enforcement saying we want a FISA letter, a national security letter that says "Turn on the mic for Leo. Don't tell anybody."

Steve: Right, right, right. Okay. So as we said at the top of the show, holding our breath. Seven days from now, March Patch Tuesday. And so we'll probably give that a little more attention than we have because there are a number of zero-days that are being fixed, and who knows what else? And I don't know if we'll ever understand how February got missed, but it did.

We did get some clarification from Amazon about what happened last week. As we know, you guys were talking about it on MacBreak Weekly before last week's Security Now! podcast. And Amazon had a problem with S3. The details, I think, are really interesting. Amazon in their formal post-action report said: "We'd like to give you some additional information about the service disruption that occurred in the Northern Virginia Region on the morning of February 28th. The Amazon Simple Storage Service (S3) team was debugging an issue causing the S3 billing system to progress more slowly than expected." Okay, so they were doing some maintenance.

"At 9:37 a.m. Pacific time, an authorized S3 team member, using an established playbook, executed a command which was intended to remove a small number of servers for one of the S3 subsystems that is used by the S3 billing process. Unfortunately, one of the inputs to the command was entered incorrectly..."

Leo: Pseudo rm-rf* - something like that; right?

Steve: Yeah, well, a typo, "...and a larger set of servers" - and apparently it was very much larger - "...set of servers was removed than intended." So they have a command-based sort of meta management system that manages this vast infrastructure. And so, like, rather than going over to servers and talking to them individually, they just do something. And, for example, I've seen that when GRC has been under attack, and it's just like, okay, we're being blasted with ridiculous amounts of traffic. I talk to Level 3, and I say, okay, just blackhole the IP. And so someone at a single console types a command, and all of Level 3's routers at all of their borders update their routing tables to remove that IP.

Leo: That's the power of BGP; right? I mean, we've seen people scrub BGP before.

Steve: Right. But it also typically, and certainly this is the case with Level 3, they have their own, you know, they can't have that happen by mistake. There's got to be an audit trail. You have to have authorization to do that. So there's like a management layer on top.

Leo: They should never have hired that guy from PricewaterhouseCoopers. That was - just because he was out of work, come on.

Steve: So Amazon says: "The servers that were inadvertently removed supported two other S3 subsystems. One of these subsystems, the index subsystem" - whoops, that sounds important - "manages the metadata and location information of all S3 objects in the region. This subsystem is necessary to serve all GET, LIST, PUT, and DELETE" - which is, you know, pretty much everything, all the critical verbs for getting and putting, listing and deleting stuff. "The second subsystem, the placement subsystem, manages allocation of new storage" - like where should we put stuff - "and requires the index subsystem to be functioning properly to correctly operate. The placement subsystem is used during PUT requests to allocate storage for new objects.

"Removing," they write, "a significant portion of the capacity caused each of these systems to require a full restart. While these subsystems were being restarted, S3 was unable to service any requests. Other AWS services in the US-EAST Region that rely on S3 for storage - including the S3 console, Amazon Elastic Compute Cloud (EC2) new instance launches, Amazon Elastic Block Store (EBS) volumes when data was needed from a S3 snapshot, and AWS Lambda - were also impacted while the S3 APIs were unavailable."

So Amazon's been building this system over time, and they've been building it carefully, and it's sort of their new services are aggregated and use the older ones. And it happens that S3 was where this all began. This is like the root core on top of which all these other things are built and depend.

So Amazon says: "S3 subsystems are designed to support the removal or failure of significant capacity with little or no customer impact. We build our systems with the assumption that things will occasionally fail, and we rely on the ability to remove and replace capacity as one of our core operational processes. While this is an operation that we have relied on to maintain our systems since the launch of S3, we have not completely restarted the index subsystem or the placement subsystem in our larger regions for many years. S3 has experienced massive growth over the last several years, and the process of restarting these services and running the necessary safety checks to validate the integrity of the metadata took longer than expected."

So essentially this is something that they knew worked, but they hadn't done for a long time. And it sounds like they were surprised that it didn't - that this was an area that wasn't scaling linearly; it was scaling more exponentially. So that they went "oops" when they realized that they'd inadvertently shut down much more of this than they expected. And they're not explaining why, but essentially it required a full restart. And then they realized with some chagrin that, oh, is it restarting? It's like, well, yeah, it's trying. I mean, it is. But it turns out it, like, took way longer than they had expected. And it's something they hadn't done for a long time.

So they said: "The index subsystem was the first of the two affected subsystems that needed to be restarted. By 12:26 p.m. Pacific time, the index subsystem had activated enough capacity to begin servicing S3 GET, LIST, and DELETE requests." But remember not PUT because that's the other thing. "By 1:18 p.m. Pacific Standard Time, the index subsystem was fully recovered, and GET, LIST, and DELETE APIs were functioning normally. The S3 PUT API also required the placement subsystem." And remember that that relies on the index subsystem, so that had to come later. "The placement subsystem began recovery when the index subsystem," they write, "was functional and finished recovery at 1:54 p.m. Pacific Standard Time. At this point, S3 was operating normally. Other AWS services that were impacted by this event began their recovery. Some of these services had accumulated a backlog of work during the S3 disruption and required additional time to fully recover."

Anyway, I just think this is a fascinating look. And it goes on, for anyone who wants more details. I'm not going to bother going into it here, but I have it all in the show notes. Just sort of an interesting look into, like, the scale of this kind of system, where first of all the design is cool, the idea that - and we know how redundant it is because, for example, you pay more for more redundancy, and you pay less for lower redundancy. And even so, you don't pay much.

I use S3. It is my, as I said before, it is my primary cloud-based backup system. And every month I get a bill. I just keep pouring stuff into it. All the podcasts are archived there, lots of images of critical subsystems and systems are archived there. I get a bill for two bucks. It's incredible. I mean, I'm not doing massive bandwidth transit because that's where you start to pay the bills. But $2 a month. And I think about all the huge amount of storage that I use. And it's incredibly robust. As we've seen, not absolutely unable to fail.

But I should just follow up and say that this was a lesson for them. And as this post continues, they assure us that many steps have been taken as a consequence of what they learned. It is no longer possible to enter that typo. There's much more oversight.

There's another system that watches what the impact will be and says, whoa, wait a minute, is this what you meant? And so there's, like, much more accident prevention so that the tool, which arguably had become overly powerful - and this is another one of those things where, over time, the very, like the birthing assumptions no longer really held; but no one had revisited them because there were other things to do, forward progress and work. So, anyway, just an interesting case study, I think, for this kind of massive, super-enterprise scale and how it operates, how it's resilient, and how it gets managed.

Leo: I thought it underscores, just as Cloudbleed did last week, the problems with a monoculture.

Steve: Mm-hmm.

Leo: It wouldn't have been such a big deal except that everybody uses S3 and Amazon Web Services.

Steve: Yes, exactly, a very good point. So suddenly, wham, you know. And in fact, I just saw that because we'll be talking about an interesting site viewer tool later. And someone sent me a link to it. Actually it was our friend Simon Zerafa sent me a link that he had run GRC against it, which is kind of a very boring response because GRC is a small site. And I looked at their list of recently run scans and clicked on CNN, and it was just, like, full of AWS stuff. And I thought, ooh, CNN would have been impacted if there wasn't redundancy.

Now, the flip side of that story is us delaying the podcast last week because my Internet connection was down for a couple hours. And I got a lot of people who were like, what? You mean you don't have an alternative emergency Internet connectivity backup of some sort? And I said, okay. My response was a little defensive, I guess. But 601 podcasts. So in 600 previous podcasts we have never had a single problem. So to me, this represents the proper degree of concern, which is almost none because...

Leo: It's just a podcast, folks.

Steve: Exactly. It's a prerecorded podcast.

Leo: People ask us the same thing. "What would happen if the power went out at TWiT?" We'd be down. "What would happen if the Internet went out at TWiT?" We'd be down. "Well, don't you have backup?" And my answer is always the same. It's just a podcast. This isn't a hospital. It's not even network television. You can wait a few hours, not a big deal.

Steve: And I do have things I consider mission critical. GRC has never been down. GRC has redundancy like crazy, I mean, where I don't even - I even have separately physical servers, and I've got hardware firewalls isolating because I just don't trust anything. I mean, I have RAID 6 on SSDs with battery backup. I mean, you cannot take it down. So, I mean, a DDoS will. But it's like, okay, again, people were like, "Oh, my god, you're under attack? Move to Cloudflare." And it's like...

Leo: Agh, my hair's on fire, my hair's on fire.

Steve: Yeah. So it's like, okay, let's just keep a sense of perspective. So, yeah. And when my bandwidth has a problem I, you know, like I said, hit the beach. It's some nice-looking beach weather right now.

Leo: Incidentally, we had a great conversation yesterday on Triangulation. I encourage everybody who listens to Security Now!, I think you'll enjoy it. Marc Rogers joined us. He's a legendary hacker. But also SecOps, he's in charge of security operations at DEFCON, has been for 15 years.

Steve: Nice.

Leo: That's a pretty good street cred. And he's the security guy at Cloudflare. So he came on. We talked a little bit about Cloudflare, Cloudbleed, and how they handled that. But we also talked about his hacking history. And it was a great conversation. I think you'd enjoy it, too, Steve. One of the things he said, we were talking about NSA surveillance, and I said, "Well, wouldn't the NSA have every outbound call to Russia that anybody made from anywhere during the election?" And he said, "Well, they might have collected it, but they don't have the capacity to store everything."

He said there's no doubt they collect everything. He used to work at Vodafone, at British Telecom, for 10 years. So he kind of has a knowledge of what is going into the government databases. He said, you've got to understand, there's so much data, even with this new Utah datacenter, he said, they might have a few weeks or a month, but they can't store stuff forever. They have to pick and choose and say, well, we'll save this stuff, and this stuff's going to get overwritten.

Steve: Right. And in fact we know from the early Snowden revelations, where we were looking at the nature of the equipment installed in those closets in the major Internet interchange nodes, that they are, in the case of text, they are keyword search sorts of things. So you put filters on certain traffic and say, you know, capture streams that contain the following. So I wouldn't be at all surprised if there is speech recognition stuff at some level where it's like, okay, just look for these words in a conversation. And if you see them, then that we want to store. Otherwise, we don't have infinite storage.

Leo: Yeah. He was good. He also is a car hacker and talked about Tesla and hacking the Tesla. This guy is, I think, pretty good at what he does. So recommend that, yesterday's Triangulation episode.

Steve: Nice. So, okay. And then some other people, responding to our discussion of Cloudflare last week, one in particular, I just found one, I just pulled one tweet out of many from someone who called himself Amateur, who said: "@SGgrc Did you read the same response I did? Cloudflare," he writes, "totally blamed what they called 'ancient software' and acted like this didn't matter." And I got a number of responses like that, which caused me to think, okay, did I get that wrong? And then I realized that what I was relying upon much more was the clear responsibility-taking discussion that Nick Sullivan had with you, Leo, on The New Screen Savers, where he was completely forthcoming.

And what I saw of Cloudflare's written disclosure, which was written also by our friend John Graham-Cumming, it appeared to be consistent with what Nick had said. But I imagine that someone reading only a more carefully written corporate post might come away with a different feeling than someone, as I did, watching a completely candid, off-the-cuff interview where he was responding to your questions and saying, yeah, you know, this is what happened. So if anyone still has any doubts, let me encourage you to watch Nick at the beginning of The New Screen Savers from Saturday before last, that was February 25th, Episode 93. I put a link to it in the show notes. Your interview with Nick, Leo, is right at the top of the show so people won't have to dig down into it to find it, if that's the only thing they want to watch. He's right there at the beginning.

And anyone, I think, would come away with the feeling I did, which was they responded immediately. They responded responsibly and really did take responsibility for their use of this software which was actually - and where the problem was triggered by their on-the-fly replacement of that so-called "ancient software." So anyway, many people were like, wait a minute, is that the proper characterization? And the only thing I could think is that I got it directly from Nick from watching his interview with you.

Leo: And I think you'd say the same thing if you saw Marc's conversation yesterday with me. The one question I had, and it's still an open question, is, well, but do we know, was any damage done? How much damage was done? Because that's the thing that's kind of unknowable. We know that data is leaked.

Steve: Correct.

Leo: But we don't know what data, for who, for what sites. And he said, "We would expect certain kinds of traffic based on somebody getting a hold of any information that was useful. We haven't seen any evidence of that." So while you can't say everything's fine, we're obviously still watching. But so far, so good.

Steve: Right. And some listeners did point out that there are other spidery things than just the arguably responsible search engines.

Leo: Right. Well, that's true.

Steve: Which is true. Again, they would have to have known to look or have discovered this and then gone on a mission for trying to find information. And it isn't something where you're attacking a given site. It's opportunistic, completely out of your control what it is you may get. Low probability of getting anything useful. And then you would have to understand what it was.

Leo: Which is why a government is not going to be as interested as you might think because they have targets.

Steve: Right.

Leo: And so it's useless for that. The chances of you or me having information revealed is infinitesimally small.

Steve: Right. And in fact we also do know that there's been a lot of dialogue in the last few days after our President tweeted on Saturday morning that Barack Obama was wiretapping Trump Tower. The fact is that, even if there were conversations involving non-U.S. citizens, as soon as any wiretap realizes there is a U.S. citizen on the line, the rules change. So, yeah, I just don't think it's nearly - my point is that, if our government were to find information that they can't have, well, they're not able to look at it, either.

Leo: And you can't really use it in court, either.

Steve: Well, exactly, exactly. So we've discussed - we revisited last week the pending question of Amazon's response to the police warrant in the suspected murder, well, somebody was at least killed, we don't know who did it, but of this person in the hot tub, remember, from November of 2015. I thought it was 2016. It may have been...

Leo: No, no, it was two years ago, yeah.

Steve: Okay. So what happened was we're not going to get any judicial opinion about search warrants covering past audio obtained by and queries served by Amazon's Voice Assistant technology. In documents which were filed last Monday, which we didn't know about for last Tuesday's podcast, but they just came to light, the defendant in that case, James Bates, said that he was willing to allow law enforcement officials to review information contained on his Amazon Echo speaker before the company handed the data over on Friday. So he's pleaded non-guilty. He said, "Fine, I have nothing to hide. Amazon has my permission to turn over anything that they have."

So it would have been interesting because Amazon, of course, is big enough to fight this with enough strength that this might have ended up being decided by a court, maybe ultimately by the Supreme Court. We don't know. But it's not going to be this case that determines that because in this case they just essentially agreed to say, yeah, okay, fine, let them have what they want.

A group of security researchers who are with the Fraunhofer Institute for Secure Information Technology - their handle is TeamSIK, S-I-K, which sounds a little hacker-ish, but it's actually SIK stands for Security Is Key. And these are obviously the real deal. They wrote: "We meet up regularly in our spare time. Our main motivation is to work on interesting security-related projects for fun, with the goal of exposing security issues. Currently," they write, "our main issues of interest are Android applications."

Well, this was in the news last week because they examined carefully and closely, in order to obtain the level of detail they did, the top nine password managers on Android.

And those were: MyPasswords, Informaticore Password Manager, LastPass Password Manager, Keeper Password Manager, the F-Secure KEY Password Manager, Dashlane, Hide Pictures Keep Safe Vault, Avast Passwords, and 1Password. So, and the way they described this, like what they came away with was interesting.

They said: "There are different policies for the generation of secure passwords." They give us a little bit of background on password managers. "However, one of the biggest challenges is to memorize all these complex passwords. Password manager applications are a promising way of storing all sensitive passwords cryptographically secure. Accessing these passwords is only possible if the user enters a secret master password.

"At first sight, the requirements for a password manager application seem simple: storing the passwords of a user centralized in a secure and confidential way. However, how is the reality on mobile password manager applications, especially on Android? Applications vendors advertise their password manager applications as 'bank-level' or 'military-grade' secure. However, can users be sure that their secrets are actually stored securely? Despite the vendors' claims, is it nevertheless possible to obtain access to the stored credentials?

"In order to answer these questions, we performed a security analysis on the most popular Android password manager applications from the Google Play Store based on download count. The overall results were extremely worrying and revealed that password manager applications, despite their claims, do not provide enough protection mechanisms for the stored passwords and credentials. Instead, they abuse the users' confidence and expose them to high risks.

"We found several implementation flaws resulting in serious security vulnerabilities. Some applications stored the entered master password in plaintext, or implemented hard-coded crypto keys in the program code. Consequently, attackers can easily" - now, that's a little questionable because I looked at this carefully also. But they say "attackers can easily circumvent the crypto algorithm" - so I would say "can" circumvent the crypto algorithm - "altogether and thereby gain access to all of the user's data. In other cases, we could simply access all 'securely protected passwords and credentials' with the help of an additional app. Once installed on the device, this malicious app extracts all passwords and credentials in plaintext and sends them to the attacker.

"In yet another case, we could use a so-called data residue attack to access the master key of an application. In most of the cases, no root permissions were required for a successful attack that gave us access to sensitive information such as the aforementioned master password. Furthermore, many of the apps completely ignore the problem of clipboard sniffing, meaning that there is no cleanup of the clipboard after credentials have been copied into it.

"While this shows that even the most basic functions of a password manager are often vulnerable, these apps also provide additional features which can, again, affect security. We found that, for example, auto-fill functions for applications could be abused to steal the stored secrets of the password manager" - and we've been talking about form auto-fill hacks and attacks in the last couple podcasts - "using hidden phishing attacks."

So they end up in nine password managers, finding 26 different vulnerabilities. All of the ones they disclosed have been fixed beforehand. Avast is the only one that had, first of all, many problems, many more than any of the others. And about half of them remained unfixed. And I should mention that this all happened back in August and September of last year. So there's been plenty of time for these to get fixed. All of the other password managers, including Last Pass, have had those problems which - so this was all responsibly disclosed and responsibly handled.

In the case of LastPass, which I was most curious about, they write: "The Android app of LastPass comes with an access control mechanism which prohibits arbitrary usage of its functionality. By default, the user is asked to enter his master password in order to gain access to the application. Due to its enforced complex requirements" - meaning LastPass Android requires a complex master password - "a user can easily get frustrated entering the master password over and over again. Therefore, LastPass offers to substitute the master password with a PIN mechanism." So here, again, we're trading security for convenience. "Thereby the user can agree to save the master password on the device" - because remember, the master password is what decrypts the store, that is, the storage blob, so you still need the master password somewhere.

So LastPass offers to substitute the master password for a PIN. "Thereby the user can agree to save the master password on the device and shift all access control to a PIN, which can be from 4 to 14 digits long. The master key and the PIN are symmetrically encrypted and stored in a shared preferences file in the local app folder. The key and PIN are stored encrypted. The key for encrypting and decrypting the credentials is hard-coded into the application's source code." Thus the problem, and that's what LastPass - that's what these guys found. And LastPass said, okay, you're right, we could do better. So the key for encrypting and decrypting the credentials was found to be hard-coded into the application's source code.

However, even so, for stealing the encrypted master key or PIN, we assume, and that is they required, that the attacker gained physical access to the device, which we're now calling the Evil Maid attack, and in which case that approach required reading out the LPAndroid.xml content, which did not require a root exploit. But not on each Android version. So only on some was this possible. The second approach did require a root exploit, which exists for various types of Androids, depending upon version.

So it wasn't, again, so I wouldn't call that easy. That required either physical access or a root exploit that then allowed you to get it. But even so, LastPass could have done better. And before this was made public, actually quite some time ago, this got fixed. And so I only looked at that level of detail at LastPass. But I've got all the links in the show notes for anyone who might be using - for example, I know 1Password was also popular. They had a similar - there were three problems that all sort of circled around this for LastPass, three different discrete vulnerabilities. I think 1Password had the same. And again, they were all fixed.

So I just say props to these guys for taking a careful close look at these. And it really looks like what we're seeing is examples where the manufacturers do what they think is, you know, they're trying to implement the best security they can, while also keeping their devices sufficiently easy to use. And even so, a third party is still super useful to, essentially, to challenge those assumptions and say, okay, this could be better over here; and then for the original developer to say, uh, yeah, you're right.

And of course I have the perspective of being working on SQRL. And so whenever I see any of these things, I challenge myself, too. I go, okay, wait a minute. How have I handled that? And because I have some of the same sorts of tradeoffs that, for example, in my implementation of SQRL, although this is not part of the protocol, this is part of the how you should do it right, I have handled all of this in a way both with entropy, harvesting, and solving the problem of needing something very complex as a get-out-of-jail-free card that we call a "rescue code," yet needing something interactively simple; yet at the same time not leaving debris around where, if someone got a snapshot of it, they would get any substantial benefit. So, I mean, these are hard problems to solve, and all you can do is the best possible. In some cases, what is being seen is that manufacturers or software developers have not done everything they possible could.

We discussed a few months ago a file-free malware that was - in fact, it was Kaspersky who actually discovered this in their servers, which their scanners weren't finding because they were scanning the hard drive. They were not looking in RAM. And so they discovered a RAM-resident malware, which was the first time that we had seen that. Well, we've now found another one. Cisco's Talos security research group discovered a new and relatively sophisticated attack which launches from an email-phishing malicious Word document and successfully executes a Windows PowerShell backdoor by communicating with command-and-control servers through a series of - get this - DNS requests.

So we know about DNS. You ask for a machine dot domain name, and you receive whatever is in this DNS record. But you're able to ask, for example, you could do a forward lookup, that is, give me the IP address of this machine and domain. Or you could do a reverse-lookup. You can also ask for other kinds of records, rather than just an address record. And in this case the system uses text records, TXT records, which is another type of thing you can store in the database. And they're increasingly used for various things. The SPF, Sender Provider Framework, which is used as an antispam system, uses DNS TXT records. So whereas originally they were rare, they're becoming quite common now.

So Cisco's dump of this work that they did was called "Covert Channels and Poor Decisions: The Tale of [what they're calling] DNSMessenger." And they go through it in detail. I won't take us all through it. I do have a lot more detail in the show notes. But essentially they're using a VB script that runs from Word, which a user is tricked into opening, in order to use PowerShell, which Microsoft has continued to make more and more powerful and capable. Basically, it's full scripting of Windows admin tasks. Anything you could imagine wanting to do, you can now do in PowerShell. It's multilevel, hierarchical, object-oriented, insanely powerful.

And what's tricky about this is that this avoids antivirus. It gets past currently the majority of AV because they're not looking for this. And it uses DNS, which has generally been regarded as safe. And the malicious content is obtained over the 'Net, through DNS TXT records. So there's nothing malicious in the code that just sort of benignly says, oh, go fetch me a TXT record at this domain. So that's something you can easily do from PowerShell. Unfortunately, what it receives is a blob which, when decrypted, does become malicious. And it turns out that there is a way, then, using PowerShell, to create persistence through the registry because you're able to create a backdoor using Windows Management Instrumentation, the WMI database. So that allows you to create something persistent which can survive across reboots. So no actual files written by the malware; yet, once this thing gets into your system, if you make the mistake of opening the document, your system can be infected just over DNS. Wow. The bad guys are clever.

Leo: But reboot would fix it; right?

Steve: No, it survives reboot.

Leo: Well, that's not nice.

Steve: No. No. No, and it's...

Leo: So it must write something out. How does it survive reboot?

Steve: It's in the registry. It uses the Windows Management Instrumentation.

Leo: Ah. So it doesn't have a file, but it does store itself on the file system.

Steve: Correct. Ultimately it does, right.

Leo: Well, they're just getting so clever. I've seen - but that's what Mirai was. It was RAM-based only; right? Because if you rebooted the router, Mirai was wiped.

Steve: Yes. So Chrome has further tightened up its security on macOS, actually bringing it to parity with some features that already existed in Chrome for Windows. Chrome has their safe browsing initiative. In this case it's broadening its protection of macOS devices, enabling safer browsing experiences by removing defenses against unwanted software and malware which targets macOS. So as a result, now and in future, macOS users using Chrome will start to see more warnings as they navigate to dangerous sites or download dangerous files.

As part of this next step towards reducing macOS-specific malware and unwanted software, which we've begun to see a little bit of, safe browsing is focusing on two common abuses of browsing experiences: unwanted ad injection, and manipulation of Chrome's user settings, specifically the start page, the home page, and the default search engine. Of course, Windows users have fought with changes to their browser for a long time. Google feels and believes users deserve full control of their browsing experience, and unwanted software policy violations hurt that experience.

So what they've done is they've created an API to give extensions control of the Chrome settings, yet to not allow this to happen behind the scenes. So confirmation boxes have been added that users will start seeing to essentially confirm and prompt to allow changes to be made where those things previously could just be made behind the scenes. And that will - actually, it's not already. I misspoke. It's starting with March 31st. So the end of this month, Chrome and safe browsing will warn users about software that attempts to modify Chrome settings without permission of the user. So that's, again, sort of marching forward a little bit to make sure that not too much changes at once, and people are not inconvenienced.

And similarly, from Mozilla on the Firefox front, they have moved their Containers from their very early testing over to the Test Pilot versions. About a year ago, last summer, Tanvi Vyas, who is a security engineer, posted about contextual identities on the web. And what she wrote was perfect as sort of to give us an understanding of what they're talking about, what these Containers are.

She wrote: "The Containers feature in Firefox Nightly enables users to log into multiple accounts" - okay. So users log in to multiple, like, Internet web accounts - "on the same site simultaneously and gives users the ability to segregate site data for improved privacy and security."

She writes: "We all portray different characteristics of ourselves in different situations. The way I speak with my son is much different than the way I communicate with my coworkers. The things I tell my friends are different from what I tell my parents. I'm much more guarded when withdrawing money from the bank than I am when shopping at the grocery store. I have the ability to use multiple identities in multiple contexts. But when I use the web, I can't do that very well. There is no easy way to segregate my identities such that my browsing behavior while shopping for toddler clothes doesn't cross over to my browser behavior while working. The Containers feature I'm about to describe attempts to solve this problem: empowering Firefox to help segregate online identities in the same way I can segregate my real life identities.

"With Containers, users can open tabs in multiple different contexts: the personal context, the work context, the banking context, and the shopping context. Each context has a fully segregated cookie jar, meaning that the cookies, indexedDB, localStorage, and cache that sites have access to in the Work Container are completely different" - disjoint, separate - "than they are in the Personal Container. That means that the user can log into their work Twitter account on Twitter.com in their Work Container, and also log into their personal Twitter on Twitter.com in their Personal Container. The user can use both accounts side-by-side in tabs, simultaneously. The user won't need to use multiple browsers, an account switcher, or constantly log in and out to switch between accounts on the same domain.

"Note that the inability to efficiently use contextual identities on the web has been discussed for many years. The hard part about this problem is figuring out the right user experience and answering questions like: How will users know what context they are operating in? What if the user makes a mistake and uses the wrong context? Can the user recover? Can the browser assist by automatically assigning websites to Containers so that users don't have to manually manage their identities by themselves? What heuristics should the browser use for such assignments?

"We don't have answers to all of these questions yet, but hope to start uncovering some of them with user research and feedback. The Containers implementation in Nightly Firefox is a basic implementation that allows the user to manage identities with a minimal user interface."

So, and here is where I loved what they did. "What is the security model provided by Containers?" And this is actually in a recent post, in the last couple days, where this was part of the Test Pilot. They wrote: "The security enhancements of Containers in Nightly and Test Pilot is common across both versions and are based on a modification to the browser's Same-Origin Policy (SOP)." Okay, now, as we know on the podcast, the Same-Origin Policy, which we talk about often because it's such, I mean, you could argue it's the most crucial security operational guarantee that browsers enforce. Which ensures that documents and data, and that includes queries made by script running on a page from distinct origins, meaning like domains, are isolated from each other. It is a critical browser security mechanism, as we know, that prevents content from one site from being read or altered by another, potentially malicious, site.

So getting a little more specific: A page's origin is defined by the tuple composed of the scheme - that's typically http or https, but could also be smb:// or ftp: or whatever. So that's the scheme. That's the first part of the tuple. The host, which is the name www.grc.com, for example, and the port, so all three of those - the scheme, the host, and the port. In the case of https it's typically 443. But as we know, the URL can have a colon and a port override, so those are all bound together. And that collection, that tuple, defines the origin. So a page has that origin made from that tuple, which means that only the things on the page - the things on a page are only able to access content at the same origin, thus the same-origin policy.

So what Mozilla has cleverly done is to add an additional parameter, which they call the userContextID integer, to the normal scheme, host, and port. So now it's userContextID, scheme, host, and port. So what's so clever about this is they get to reuse all of the existing containment enforcement that is core in the browser in order to create essentially contextual instances. When a user visits Gmail in a Work Container tab, the browser performs the same-origin policy check against, for example, 2 - two as in the userContextID - https, mail.google.com, 443. That's the tuple that defines the same origin, or that defines the policy and the origin for that page. Then when the user visits Gmail in a Personal Container tab, that tuple would be 1, https, and then mail.google.com and 443. So even though it's technically the same site, the addition of this integer to the tuple allows the browser to segregate those sessions, essentially.

So those containers which are implemented with this additional integer index, separate all of the things that matter - the cookies, the local storage, the indexedDB, and the cache data - from each other and from the default container in Firefox. So that is now going to be - that's now in the Test Pilot version. And it's been in Nightly now for a while. It's instrumented. They're watching to see how people use it. So it's sort of an advanced and persistent version of Incognito, where rather than not - where Incognito mode typically works by creating a "never write anything to disk, keep everything in RAM," so that when we flush, it gets completely flushed. This essentially solves the problem that many of us have solved by using different browsers because at this point everything in a browser is global to the browser. Which is convenient when you want to have, like multiple tabs to the same site, and you stay logged on. It's inconvenient when you, for whatever reason, do want to appear as someone different.

And I'll note also that this is an issue that we've looked at, again, in the SQRL context. We've had extensive similar discussions because one of the things that SQRL does is give you a fixed identity for a website. But then the question is, okay, but what if I, for whatever reason, don't want to be me at that website at the moment? And so we've incorporated that into the design, and we call them "alternative IDs," where it's trivial to create an alternative ID which has all of the properties SQRL is known for - no trackability, no way to, I mean, essentially, you look like an absolutely unconnected, completely different person form the standpoint of the SQRL identity.

And so if you coupled that, for example, with this kind of container management, it solves the whole problem. You could be logged in as two different people, both through your single SQRL identity, yet be using an alternative, and just jump around with containers. The way they've implemented this is under the file tab there is, like, Open New Window, Open New Tab, Open Incognito Session. They've added the ability to access containers, to define or open containers. And they show you which one you're in by color-coding the container's name and right-justifying it in the URL region so you can always see if it's your work, your personal, or whatever. So nice movement for Firefox. I'm glad to see that. And looks like it's going to be convenient.

At last month's RSA 2017 conference, the team of security researchers from Cylance gave a live demo of Cylance's UEFI Ransomware proof of concept. So that's, like, as bad as it gets, the idea that something that you could get over the Internet could break through all of the multiple levels of security we have and alter the firmware on your motherboard to create persistent malware. In the demo that they showed, they successfully exploited a system with an Intel Skylake CPU running Microsoft Windows 10 Enterprise, that's build 1607, with all updates installed. They enabled all available security features in the system, including Secure Boot; Virtual Secure Mode, that's VSM; and the Device Guard under its default policy. They, like, turned everything on.

Despite all of those protections and mitigations, based on the various virtualization technologies in Windows 10, specifically designed to prevent this from happening, the systems remained vulnerable to UEFI-based or UEFI-targeted attacks through system management mode. We talked about this about a year ago, how it was possible to use system management mode in order to bypass some of the protections. Because system management mode allows direct access to physical memory, it's possible to bypass the virtualization layer of isolation which has been put in there for protection, which allows an unprivileged application to install persistent malware like ransomware, which they have done in their demo, invisibly into a motherboard's firmware.

So at the end of this month, at the upcoming Black Hat in Asia, which will be occurring in Singapore, they will be going - what they did last month was give a full live presentation demo. At the end of this month, they'll be disclosing all of the details of the vulnerabilities they exploited in their talk last month. So the bad news is, I mean, again, here's another perfect example of attacks never get weaker, they only get better and stronger. And so they have taken the concept and brought it to a full live working demo that manages to cut through all of the mitigations and protections which have been put in place to prevent that.

We've talked, and were just talking last week, in fact, about return-oriented programming, where it's possible to use snippets from the end of, or typically the tails, of existing kernel-privileged code to make things work. The details are, well, we've talked about it enough that there's really no need to go into any deeper level of detail. But one of our listeners sent me a link and said, hey, you know, what do you think of this? I've got it in the show notes. It is on GitHub.io, called Gargoyle, G-A-R-G-O-Y-L-E, a memory scanning evasion technique. And what's nice about this is it is a beautiful, extremely clear, yet very technical, I mean, like to the level we don't need to do on this podcast. But I imagine some of our listeners would be interested.

So it's a very nicely written up, detailed example of a practical working ROP, Return-Oriented Programming, exploit on Windows that demonstrates how an attacker can arrange to execute pure non-executable data. And so this person, again, we've already in the past years discussed this in many different instances and contexts. So there's no need to dig in any deeper. But I thought, hey, this is a perfect example, if somebody really wants to go real-world. I mean, there is proof-of-concept code in this posting. You can do it yourself. You can make a dialog box pop up from something that is just data. So the person who put this together pushed it all the way to here's how this ROP, Return-Oriented Programming, works in practice. So if anyone's interested, I refer them to this Gargoyle on GitHub.io, and the link is in the show notes.

So an interesting site. And Leo, you probably want to put TWiT into this.

Leo: Okay.

Steve: It's URLScan.io. It's in beta. U-R-L-S-C-A-N dot I-O. And when you just go there, you will see, as many of these sites now do, SSL Labs does the same thing, showing you the most recent domains that have been scanned. This will show you the most recent things it's seen. And what it does is it's very convenient because you can give it a URL, and it will go, pull the page, and then summarize in aggregate what the page did - how many domains it reached out to, how many requests it made, how many ads were blocked, if there was any malicious content that it tried to fetch, what percentage of it was secure versus insecure. Was any IPv6 pulled? What about subdomains? What about discrete IPs? What about what countries were fetched from, the size of the content, how many cookies and so forth.

So it really, I mean, it formats it beautifully. You can drill down. You can get additional detail. Those words across the top in green, those are tabs. So you can look at the various tabs to drill into various details. So I imagine it's something that our users will get a big kick out of: URLScan.io. And I did it for CNN, and also for GRC. As I said, GRC was, just because it's a small site that really doesn't refer to any other third parties, it's pretty small. And there you've got yours for TWiT up on the screen, Leo.

Leo: Yeah, but I didn't realize we were doing so much stuff.

Steve: Yeah, it's very revealing.

Leo: Yeah. A lot of this is Amazon load balancing. So it says, for instance, we're pulling from three different countries. Well, I don't know why. Canada, the U.S., and it looks like the U.K. But that's probably Amazon; right? I don't, you know, I don't - geez.

Steve: Yeah.

Leo: Fifteen ads blocked is pretty amazing since there's only two ad blocks on the whole page. I don't know what they're talking about there. Interesting.

Steve: Can you click on it?

Leo: No, it's not live.

Steve: How about the word...

Leo: No, that's HTTP. I don't know - you can't click on this. I'm not sure what they're getting about ads blocked. But I'm sure it makes sense.

Steve: Another cool site.

Leo: Yeah, really neat. URLScan.io.

Steve: So Matthew Green answered a question which followed from a tweet that BuzzFeed's editor Miriam Elder posted. She tweeted, on March 3rd: "Possibly stupid question. Is the Signal desktop client as secure as the mobile app?" And so Matthew wrote: "No, this is not a stupid question. Actually, it's an extremely important question; and, judging by some of the responses to this tweet" - that is, to her tweet - "there are a lot of people who are confused about the answer."

He writes: "Since I couldn't find a perfect layman's reference anywhere else, I'm going to devote this post to providing the world's simplest explanation of why, in the threat model of your typical journalist, your desktop machine is not very safe; and, specifically, why you're safer using a modern mobile device and particularly an iOS device than just about any other platform."

He writes: "A brief caveat: I'm a cryptographer, not a software security researcher. However, I've spent the past several years interacting with many folks who live in this space. I'm pretty confident that they would agree with this advice."

"So what's wrong with my laptop/desktop machine? Sadly," he writes, "most of the problem is you. If you're like most journalists and really most professionals you spend less than 100% of your time thinking about security. You need to get work done. When you're procrastinating from work, you visit funny sites your friends link you to on Facebook. Then you check your email. If you're a normal and productive user, you probably do a combination of all these things every few minutes, all of which culminates in your downloading some email attachment and (shudder) opening it in Word." Which of course were just talking about, how that could easily contain an exploit that would infect your system.

And then he cites an older post from someone named Eva from September 12, 2016, saying: "You can't tell journalists 'just don't open attachments.' They will ignore you. Journos open attachments from strangers for a living."

And so he continues: "Now, I'm not trying to shame you for this. It's perfectly normal, and indeed it's necessary if you want to get things done. But in the parlance of security professionals, it also means you have a huge attack surface. In English, this means that, from the perspective of an attacker, there are many different avenues through which to compromise your machine. Many of these aren't even that sophisticated. Often it's just a matter of catching you during an unguarded moment and convincing you to download an executable file or an infected Office document. A compromised machine means that every piece of software on that machine is also vulnerable." Okay, that was key: "A compromised machine means that every piece of software on that machine is also vulnerable.

"If you don't believe this works, head over to Google and search for 'Remote Access Trojans.' There's an entire commercial market for these products, each of which allows you to remotely control someone else's computer. These off-the-shelf products aren't very sophisticated. Indeed, most require you to trick your victim into downloading and running some executable attachment. Sadly, this works on most people just fine. And this is just the retail stuff. Imagine what a modestly sophisticated attacker can do."

So then he asks himself the question, or has this hypothetical journalist saying, "I do some of those things on my phone, as well. Why is a phone better?" He writes: "Classical desktop and laptop operating systems" - and I love what he wrote here. I thought this was very insightful - "were designed primarily to support application developers. This means they offer a lot of power to your applications. An application like Microsoft Word can typically read and write all the files available to your account. If Word becomes compromised, this is usually enough to pwn you, in practice. And in many cases, these applications have components with root or admin access, which makes them even more dangerous.

"Modern phone operating systems like Android and iOS were built on a different principle. Rather than trusting apps with much power, each app runs in a sandbox that mainly limits it to accessing its own files. If the sandbox works, even a malicious application should not be able to reach out to touch other apps' files or permanently modify your system. This approach, combined with other protections such as in-memory code signing, hardware secret storage, and routine use of anti-exploitation measures, makes your system" - meaning your mobile operating system - "vastly harder to compromise." He wrote "vastly harder to compromise."

"Of course, sandboxing isn't perfect. A compromised or malicious app can always access its own files. More sophisticated exploits can break out of the sandbox, typically by exploiting a vulnerability in the operating system. Such vulnerabilities are routinely discovered and occasionally exploited. The defense to this is twofold: First, run a modern, up-to-date OS that receives security patches quickly; and, two, avoid downloading malicious apps. Which brings me to the main point of this post: Why use iOS?"

He writes: "The fact of the matter is that, when it comes to addressing these remaining issues, Apple phone operating systems on iPhones and iPads simply have a better track record. Since Apple is the only manufacturer of iOS devices, there is no middleman when it comes to monitoring for iOS issues and deploying iOS security updates. This means that the buck stops at Apple, rather than with some third-party equipment manufacturer. Indeed, Apple routinely patches its operating systems and pushes the patches to all supported users, sometimes within hours of learning of a vulnerability, something," he says, "that is relatively rare at this point in any case," meaning vulnerabilities.

"Of course, to be fair, Google has also become fairly decent at supporting its own Android devices. However, to get assurance from this process you need to be running a relatively brand new device, and it needs to be manufactured by Google. Otherwise, you're liable to be several days or weeks behind the time when a security issue is discovered and patched if you ever get it. And Google still does not support all of the features Apple does, including in-memory code signing and strong encryption.

"Apple also seems to do a relatively decent job at curating its App Store, at least as compared to Google. And because those apps support a more modern base of phones, they tend to have access to better security features; whereas Android apps more routinely get caught doing dumb stuff for backward-compatibility reasons.

"Finally, every recent Apple device, starting with the iPhone 5S and up, also includes a specialized chip known as a Secure Enclave Processor. This hardened processor assists in securing the boot chain, ensuring that nobody can tamper with your operating system. It can also protect sensitive values like your passwords, ensuring that only a password or fingerprint can access them. A few Android phones also offer similar features, as well. However, it's unclear how these devices are implemented in contrast to Apple's Secure Enclave Processor. It's not a bet I would choose to take."

And finally he says: "So does using iOS mean I'm perfectly safe?" He says: "Of course not. Unfortunately, computer security today is about resisting attacks. We simply don't quite know how to prevent them altogether. Indeed, well-funded attackers like governments are still capable of compromising your iOS device, and your Android, and your PC or Mac. Literally the only question is how much they'll have to spend doing it.

"Here's one data point. Last year a human rights activist in the UAE was targeted via a powerful zero-day exploit, likely by his government. However, he was careful." And we covered it on this podcast. "Instead of clicking the link he was sent, the activist sent it to the engineers at Citizen Lab, who reverse-engineered the exploit. The resulting 35-page technical report by Lookout Security and Citizen Lab is a thing of terrifying beauty," writes Matthew. "It describes a chain of no less than three previously unpublished software exploits, which together would have led to the complete compromise of the victim's phone" - and I'll add, had he simply clicked the link in that text message he received.

"But such compromises," Matthew writes, "don't come cheap. It's easy to see this kind of attack costing a million dollars or more. This is probably orders of magnitude more than it would cost to compromise the typical desktop user. That's important. Not perfect, but important."

So, finally, "You're telling me I have to give up my desktop machine? Not at all. Or rather, while I'd love to tell you that, I understand this may not be realistic for most users. All I am telling you to do is to be thoughtful. If you're working on something sensitive, consider moving the majority of that work and communications to a secure device until you're ready to share it."

And I'll just mention, I've said on this podcast before, there are instances where I receive things on my desktop that I will not click. I will not open them. I truly wait, or go to an iPad, and I do them there. Because I already have internalized and understand exactly what Matthew is saying here. And he says: "This may be a bit of a hassle, but it doesn't have to be your whole life. And since most of us already carry some sort of phone or tablet in addition to our regular work computer, hopefully this won't require too much of a change in your life. You can still use your normal computer just fine, as long as you're aware of the relative risks. That's all I'm trying to accomplish with this post.

"In conclusion, I expect that many technical people will find this post objectionable" - well, I don't - "largely because they assume that, with their expertise and care" - see, okay, yeah - "they can make a desktop operating system work perfectly safely. And maybe they can. But that's not who this post is addressed to. And of course this post still only scratches the surface of the problem. There's still the problem of selecting the right applications for secure messaging - for example, Signal and WhatsApp - and finding a good secure application for note-taking and document collaboration and so on. But hopefully," he concludes, "this post at least starts the discussion."

And I'll just add something that he didn't, but this is a topic we discuss all the time, is it is exactly those very limitations of our mobile OSes that we most chafe at. I mean, iOS is more limiting than Android. And so many people choose Android because it gives them more freedom, more flexibility. It lets them have more power and do more what they want to. iOS is too restrictive. But iOS, as a result of being more restrictive, gives you more security. It's the tradeoff.

And of course, as we know, our desktops are the least restrictive of all. That's why I, even though I'm as expert as I could be on being careful, I don't know when I click a link what will happen. And so if I don't have confidence in where that link came from or what it might do, I don't do it on my desktop. Despite all of the protections that I have in place, I use something that I think is less exploitable. And that's my iPad. So I thought a really, really nice post, and I thank Matthew for taking the time to put that together. I hope people who should see it will see it. And I know that our listeners will appreciate it. Basically, it backs up all of the policies and practice and experience that we've been talking about in this podcast for a long time.

Leo: All right, Steve. Are you caffeinated?

Steve: I'm recaffeinated, yes.

Leo: Yay.

Steve: As opposed to decaffeinated. I don't want to be decaffeinated, but I want to be recaffeinated.

Leo: Recaffeinate.

Steve: So I got a kick out of this. Matthew Clayton sent a tweet saying: "Loved this detail from the New Yorker's piece on Russian cybercrime." And he took a picture of a chunk of the newspaper, physical newspaper column. And I liked, well, first of all - and he highlighted one chunk of it. He said: "Russian hackers accomplished a feat that Pentagon officials considered almost impossible: breaching a classified network that wasn't even connected to the public Internet. Apparently, Russian spies had supplied cheap thumb drives, stocked with viruses, to retail kiosks near NATO headquarters in Kabul, betting correctly that a U.S. serviceman or woman would buy one and insert it into a secure computer."

And of course some years ago we covered a similar sort of exploit where some security researchers just wanted to sort of see if people would plug unknown thumb drives, like found thumb drives, into their computer.

Leo: Of course they will. Of course they will.

Steve: Yes. And so they, like, I don't know, got 10 of them and loaded them with some tracking malware, I mean, something benign because it was for research, but then scattered them in a parking lot. Just sort of left them on the ground. And sure enough, they got pinged by that device when some unwitting person who found it on the ground thought, ooh, I wonder what's on this? Maybe there's something good here. It's like, well, uh, no, not good. But definitely something. So, yikes. So I thought that was very clever. Basically, sell inexpensive thumb drives at retail, preloaded with malware, near to somewhere you hope to compromise. And sure enough.

So many people asked, because I didn't say last week, when I was referring to monitoring my Internet connection and knowing that it was two out of three packets that were being dropped and so forth - and remember, you probably saw, Leo, because I emailed, I ended up emailing it to you and the gang, although I couldn't get it to you before the podcast because of the outage, a chart showing the packet loss history over time.

Leo: Oh, yeah, wow. It was, boy, whew.

Steve: Yeah, it was, like, not here.

Leo: Not good, yeah.

Steve: Anyway, I have been using - so I just wanted to answer the question. I've been using for years a piece of software called PingPlotter. And there's PingPlotter Free, PingPlotter Standard, and PingPlotter Pro. And unfortunately, they've been attempting to further leverage the success of their product. It should not be as expensive as, I mean, for just a simple utility, it shouldn't cost what it does. They try to sign you up on an annual contract. But there is a PingPlotter Free. It is very nice. It's a utility that I use at PingPlotter.com. And I recommend it, if you can find it for a price that makes sense to you. I mean, the free version, certainly.

And what it does is it essentially, on a timed interval, it just, you know, it sends a ping to a destination. And it also does a traceroute, so you're able to look at, by emitting packets with successively shorter time to lives, TTLs, the packets expire on their route, and that allows it to probe the path that you're going through and, like, find where is the problem where your traffic is dying. So anyway, PingPlotter.com. I like the utility a lot. I wish you could get more for less money, but that's the way they do it. And it's gotten more expensive over time. I had Ping Plotter Pro that I bought a long time ago, full feature, and I purchased it for a reasonable price. And now it's way more expensive.

Oh, and Richard Covington tweeted, he said, following up on our discussion, remember we were talking about covering the lens on your webcam versus the problem with doing it with a microphone. And he had a good observation. He writes: "One can disable the microphone on computers. Just plug in the male" - Leo, your mic's open, by the way.

Leo: Oh, sorry.

Steve: No problem. "Just plug in the male mic plug with the wires tied together." And that's like, oh, I hadn't thought of that. If you have a computer that has a microphone jack, and you'll want to verify that this works, but that is an interesting idea is to use the hardware override of the built-in microphone to the external microphone to disconnect the hardware of the internal microphone. You'd have to maybe get a little right-angle jack so it's not sticking out too far, or just the jack without any wires coming out of it. And you might want them shorted. He was suggesting shorting them so that you don't pick up any buzz or hum or anything because, if you short them, then it'll just like be grounding the microphone's input, and you'll get complete silence. So good tip, Richard. Thank you for that.

Also, Kevin Bryant actually cc'd me. So he was tweeting out to the world, saying "GRC's ShieldsUP will only scan the first 1056 ports," as I've mentioned before, 1024 plus an extra chunk. He says: "Are there no one-click, scan-all-my-ports websites?" And then he cc'd @SGgrc, which is how this came to my attention. And I'll just say that this is something I considered a lot. And I designed ShieldsUP back when it was behind my two T1s. And scanning all the ports is a much bigger job. In fact, it's a 64 times bigger job. So I'm scanning 1024, plus a little bit. But there are...

Leo: You do the canonical ports.

Steve: Correct. But he wants, Kevin is wanting to scan all 64K ports, all 65536. And so we could definitely do it, but it's 65 times, or 64 times longer and more bandwidth and more work. And I don't know what I'll do. Maybe, eventually, I mean, once SQRL's finished, once SpinRite's behind me and brought back up to speed, it's probably time to look at ShieldsUP again. Maybe switch to IPv6. And then I don't know. Maybe - I just don't know. I mean, it feels like it's too much to do for free because it would be a substantial load for a free service. So maybe...

Leo: You could use Nmap and do it yourself. I guess, no, it has to be outside your network, doesn't it, to be effective.

Steve: Right.

Leo: But you could use Nmap from work to scan your home. Or, no, I guess you couldn't because that's inbound.

Steve: Yeah. And, I mean, there's a lot that I do that people take for granted, like I figure out closed, opened, and stealth. There's a bunch of stuff I do about routers that have anti-DDoS technology in them. And it just, you know, it's easy to use. You just go to a website. So maybe I would create a little subscription service, or a micropayment. I just don't know. But it is what it is for now. You can ask it to scan any port you want. That's all free. Just not all of them at a single button.

And Simon Zerafa also sent me an interesting observation. He retweeted somebody who found an "I am not a robot" CAPTCHA that was malicious because it was fake. It was not from Google. It was invoking a malicious script. And we've talked often about how what you don't want to do is click on something malicious. But what this person realized actually by encountering it, and Simon was nice enough to forward it, is that, if we get very comfortable with the "I'm not a robot" proof, like clicking on "I'm not a robot," nothing prevents that from being spoofed. And where we're actually clicking on something that in this case activated a PowerShell payload. And we were just talking earlier about how powerful PowerShell is.

So the problem is browsers are still not, you know, they're fighting this problem of trying to keep us aware of things that might hurt us without just popping up dialog boxes all over the place so that we just get trained to dismiss them without taking them seriously. That's not a problem that seems to have a good solution at this point.

Leo: I know it's not ShieldsUP, but I just ran Nmap on my own server, and it does at least go through and look at all the open and closed ports. It doesn't show stealth. Well, closed would be closed, but it doesn't show stealth ports, obviously. But that's a good start, isn't it?

Steve: Yeah.

Leo: That's my server at home. Nmap.

Steve: And we talked about Nmap recently as a great local tool.

Leo: Yeah.

Steve: Okay, Leo. The coolest mind-blowing waste of time probably ever. I'm not kidding.

Leo: This must be good, wow.

Steve: I was a high-school freshman in October of 1970 when the October issue of Scientific American came out. And in Martin Gardener's Mathematical Games column, he introduced the world to John Horton Conway's "Life," also known as "Conway's Game of Life." What it was, was an incredibly simple cellular automata system. And it wasn't the first, but it was arguably the best. And so here's what that is. You take a two-dimensional grid, like graph paper, which is obviously composed of cells. And a cellular automata has rules that define the life and death, the binary state, the on-ness and off-ness, the living or not living state of individual cells on this grid.

And what was so perfect about what Conway did was he defined - and this was what is so fascinating about it. Very few rules, just four rules, which were easy to know and memorize and implement, that made this thing, which was a machine, made it go. So any live cell with fewer than two live neighbors, okay, so on a rectangular grid you're going to have a single cell will be surrounded by eight neighbors; right? So there's eight cells, and those are called neighbors, neighboring cells.

Any live cell, any cell that's currently alive, with fewer than two live neighbors, dies because of loneliness or underpopulation. Any live cell with two or three live neighbors lives on to the next generation. So the next iteration. So the way you think of it, you have the current state of the grid. And then you go through and you, with that state, you determine what the next state will be by processing every cell to see whether it will be alive or dead in the next, based on these rules, in the next update. Okay. So any live cell with two or three neighbors lives on because it's got just enough to be - it's not going to die of loneliness or underpopulation. It's just enough to stay happy. But any live cell with more than three live neighbors dies from overpopulation. And lastly, any dead cell with exactly three live neighbors becomes a live cell as if from reproduction. So you create life if you have exactly three live neighbors.

So, okay. So a couple simple cases. Take a 2x2 - you have the whole grid clear - a 2x2 square of cells. So two live cells next to each other, and then immediately below them two more live cells. So each one of the live cells has its three neighbors. So it gets to live. Yet because in this configuration none of the dead cells around that block have exactly three neighbors, it's only when you have exactly three neighbors that you get to spontaneously come to life. So that's called a "block" in the terminology, and it just sits there. Nothing happens. Generation after generation, there's no birth of cells around it, and they don't - and the ones that are alive, stay alive because they've got just the proper balance.

Now, take another simple example. This is known as a "blinker." It's three cells in a row. So you've got a whole empty grid with three cells in a row that are on. Okay, now, the cell in the middle of the three, it's got its two neighbors. So it stays alive. But the cells on the end, they've only got the one neighbor, that middle cell. So that's not enough. They die. Looking at the cells surrounding this little set of three, only the two cells above the center and below the center have exactly three neighbors. So that's the birth condition.

So what that does is that means in the next generation the cells on the end die, and the cells above and below the center come to life. So in other words, this thing flicks back and forth. It blinks between being a little horizontal line and a little vertical line. Okay. So there's the basics. But when you start playing with this thing, oh, my goodness. There are gliders which are able - they're a little five-cell configuration that move diagonally forever. They just fly off the graph. There are glider guns which are these amazing machines where these two things move back and forth and bounce into each other and, through some miracle, they emit a glider that then starts moving off the screen diagonally.

Oh, yeah, there's a glider gun. And, oh, it's - there are space ships which move horizontally. There are puffer-type breeders that move along, leaving glider guns in their wake. And then the glider guns, of course, begin emitting gliders from them. You have the notion of the speed of light because, of course, the speed of light is "C" in our universe, but it's an iteration. So the speed of light is a grid step because, if you think about it, nothing in the Game of Life can move faster than one cell. So then you rate the speed at which gliders glide and space - there's also space ships, I forgot to mention, space ships move in what fraction of "C" they're able to obtain.

Okay. So with all of that background, some crazy person, I can't even imagine doing this, built a digital, a fully functional operating digital clock using this cellular automaton, so that the live cells generate the digits of the time of day. And it ticks, and it has an a.m. and p.m. indicator. And it's just like, I mean, when you understand how few and simple the rules are, and the idea that this is even possible, it is just - it's just mindboggling. So again, I mean, it's got even colons. And you can't have a colon. A colon is an unstable body. A colon will immediately, the entire center will go extinct. I mean, I played with this enough as a kid that I can look at this, and I can tell you what will happen because this is the kind of upbringing I had. But anyway...

Leo: Just can't get mine to work.

Steve: And it's just an incredible piece of work. Just incredible. And I have to say, for anyone who's interested, you can find Life on every platform that has ever been created. There was probably even a Nintendo GameBoy version. Certainly you can get it on Android and on iOS and on Windows, on every platform. And so if you're curious, just play with it, as Leo has been while I've been talking about it because you obviously found one on a web page; right?

Leo: Yeah. No, well, he actually links in his post on Stack Exchange for Code Golf, he links to a simulator that's a web-based simulator. So you could paste his code into the JavaScript Conway Life Simulator. Oh, I forgot to set the generation step to 512. There we go. I've clearly got some work to do. Anyway, very cool.

Steve: Oh, and it's just a stunning piece of work. Again, utterly useless, but fabulous.

Leo: Yeah, I've talked to people who actually became researchers in AI because they started with Conway's Life.

Steve: Oh, Leo, it's just I mean, it's...

Leo: So a lot of people did; right?

Steve: ...so elegant. It's a simple set of rules. And when you look at how rich the result is from those four simple rules...

Leo: There we go, there we go.

Steve: ...that operate on a grid. Look at that thing.

Leo: This is insane. I don't know what kind of amazing cognitive ability this guy has because...

Steve: I know.

Leo: ...I don't know where you'd go to make this, or how you'd figure it out. He doesn't publish the code, does he? He just has the...

Steve: Oh, it's all - I think it's all there to be figured, yeah, to be reverse-engineered.

Leo: Well, yeah. There's a big binary blob, I mean, you know, of numbers. Very cool. I mean, just amazing. Just amazing.

Steve: Yeah. And utterly useless.

Leo: Yeah. He probably spent years of his life.

Steve: But again, hats off. Salute, you know?

Leo: Very similar to writing, for instance, a general purpose Turing machine in Minecraft, you know, that kind of thing. I've done that.

Steve: And Turing machines have been written in Game of Life, by the way.

Leo: Oh, of course, of course, yeah.

Steve: And it is Turing complete. You can solve any problem if you have enough little cells. I mean, and back, you know, I was at the AI Lab at Stanford in '72 and '73, and everybody was implementing in PDP, what was it, PDP-10 code, like the fastest implementation. I thought, some day I'll take an FPGA and do it in hardware. It's just, oh, anyway. It's just - it's a perfect piece. And so for anyone who has never been exposed to it, now you have been. I apologize for the loss of the next several weeks of your life. If it hooks you, the hook could go deep because, boy, it's just an amazing recreation. And in fact Conway had no computers at the time. He did this on a Go board.

Leo: Yes, exactly, yeah. Totally cool.

Steve: So speaking of going, I got a tweet, I guess it was a DM because it looks long enough. Jordan, @Rand0mByteZ, on March - and meanwhile your clock is ticking.

Leo: Oh, yeah.

Steve: On March 1st sent me a note, he said: "Hey, Steve," and he says in brackets, "[SpinRite Testimonial]. The CEO of my agency recently brought in her personal laptop, which held her recently created, but not filed, tax return."

Leo: Uh-oh.

Steve: "The laptop was basically on life support." And he says, "I don't even think Dr. House could have brought this thing back. The OS loaded, but you couldn't do anything. Right-clicking took over a minute to register. The laptop's hard drive was just replaced not even a year prior." But on the other hand we know that laptop hard drives tend to get subject to physical abuse if, like, someone bumps them or drops them while the drive is spinning. So that can be a problem.

He says: "But I decided to break out SpinRite. I started a scan at Level 2. The scan got to 0.14% complete and reported corrupt bits. At 0.16% I realizing that this called for the big guns. So I stopped the scan and turned it up to 11." He's joking. He says, parens, "(Started on Level 4)." He says: "I kicked off the Level 4 scan and let it run again. It got to 0.19%, but it was running very slowly. So I stopped the scan and thought to try it on Level 1. Bingo. Level 1 found and corrected the corrupt bits, and the rest of the drive was scanned, checked, and marked good.

"Rebooting the machine, the OS came up normal, and the laptop was running as if it were a new drive all over again. The CEO got to file her taxes, which made me look like a superhero. Upon payment for my services, there will be another yabba-dabba-doo coming your way. Thanks again for the best HDD recovery tool out there. I'm not sure why your competition even tried." I don't really think we have any. But anyway, Jordan, thank you for your report.

Okay. Our two final pieces. When I read this, I immediately had, like, one of those aha moments. It's like, oh, my goodness. I mean, it was so immediately clear to me that this was a problem I hadn't recognized before, and it probably will be for our listeners: the threat presented by deliberately fooling AI. AI, as we know, is all the rage now. We have crazy computing power, insane amounts of storage, and massive connectivity. What is guaranteed to happen, based on all of the history that we've seen of our adoption of technology, is that we will quickly grow to depend upon these technologies and to incorporate them into our lives, as we have with email, social networking, and most other technology that is useful and used on a daily basis. And insufficient attention will be given to the question of how it might fail, or what we'll do if it does.

The problem is everything we've learned about security suggests that things generally work well only so long as no one is trying to deliberately subvert these technologies because our technologies are still not being designed, very much as Matthew said in his blog posting, still not being designed with an eye to malicious abuse. That is, getting them working is hard enough. And we generally ship it at that point. Everyone's got a boss a couple levels up who says, "Why is this taking so long? What's wrong with this?"

So what I ran across, thanks to one of our listeners, was a link to a blog posting at OpenAI.com titled "Adversarial Example Research." They explain that "adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake." And I love this analogy. "They're like optical illusions for machines. In this post," they continue, "we'll show how adversarial examples work across different mediums, and we'll discuss why securing systems against them can be difficult."

And that's all I'll share from this. Anyone who's interested to dig in can go further. But I just loved that analogy of optical illusions. There's the famous one, maybe because it's so powerful, where you have two parallel lines, and then you present it with sort of a starburst of lines going in different directions. And your eyes are incapable of not seeing the parallel lines as being bent, as being bowed. No matter how much you tell yourself that they are still straight, they just don't look straight anymore.

And so the point is that there's an instance where - and of course optical illusions are many and famous because they trick our neural wiring into seeing something wrong. And it must be the case that AI is massively foolable. My intuition just says, oh, boy. Let's see. Where are we? We're in Podcast 602 today. So we're not quite two-thirds of the way finished with our three-digit podcast series.

Leo: By the way, Steve says he's retiring at 999.

Steve: Probably, by the time we get to 999, we will have talked about real-world instances where AI was deliberately fooled. Mark my words. To me, this feels like a worrisome slam dunk. Somebody will have created, I don't know what, something weird that someone puts on a street that causes self-driving cars to go insane because it interacts with their AI in a way that nobody thought of before; but somebody figured out that, if you have three X's in a row with a certain weird perspective somehow, the car goes off the cliff. I mean, who knows? But it feels to me like this is a very worrisome, very obvious problem that we're not going to be looking at until we see some exploitation of it, hopefully by researchers who are able to draw our attention to this before it hits the real world.

But everything we know tells us that the way this stuff is developed, the way we become dependent upon technology is kind of premature because we want it so badly; and that the technologies we end up designing are porous, that is, they're just not rigid enough. They're not resilient in the way we need. And to me, it feels like AI is really prone to having a bad reaction when something is presented to it that deliberately confuses it. I just - it feels to me like that is just going to be way too easy to do.

And, finally, the dark side of automated domain validation certificate issuance, and thus the reason I titled this podcast "Let's Spoof." I'm going to share, because it's got all the facts pulled together in a nice piece, a posting made yesterday by Vincent Lynch, who's with The SSL Store. And while you may - and I was a little suspicious of his motives, this isn't about that. I agree with him completely. So his blog posting is "A Call to Let's Encrypt: Stop Issuing PayPal Certificates." And I'm just going to cut to the chase so I don't lose anyone's attention here, to say that Let's Encrypt has issued - is everybody sitting down? Leo, are you centered over your ball?

Leo: I'm on my ball, baby.

Steve: 988.

Leo: Whoa.

Steve: PayPal certificates.

Leo: Okay. When there can be but one - the real one.

Steve: That might be a problem, yes, exactly. There's only one valid one. As he says: "Data shows widespread use of certificates for phishing. This is an open letter," he writes, "to Let's Encrypt regarding its issuance of certificates containing the word 'PayPal.' Within the SSL/Certificate Authority industry, there is an ongoing debate about SSL certificates for malicious websites. The big question is if CAs should be policing the content and nature of the sites they issue certificates to. Should CAs be filtering out and rejecting certificate requests if they believe websites will use them for phishing or malware distribution? Should CAs revoke certificates for websites that are reported and proven to be involved in these activities?"

In other words, and I'm paraphrasing here, I mean, I'm injecting my own editorial, whose problem is this? This is a problem. And in fact I should say phishing is the big worry that we've been addressing with SQRL for the last year. And we did find a solution, but we can't implement it as easily as we want to today. SQRL can protect, but we need a few more hooks that are sort of outside of SQRL's domain responsibility. But the point is this is like the big problem. And the question is, whose responsibility is it?

So continuing his post: "Prior to Let's Encrypt, all major CAs" - the clock is ticking.

Leo: I wanted to get the big turnover there. Okay, go.

Steve: Ooh, yes, nice. It still hasn't rebuilt itself, has it. "All major CAs supported the view that certificates for malicious sites should be rejected or revoked, and Let's Encrypt has stirred the pot by taking such an aggressive stance on the subject." Meaning not - somebody else's problem. We're going to just issue them as long as you can prove you own the domain, a domain validation certificate, as we've discussed.

"The Executive Director of Let's Encrypt has previously written about his view on a Certificate Authorities' responsibilities. They think it's not a CA's job to determine if the site requesting a certificate is safe or legitimate" - like it's somebody else's problem - "and that even when one tries to, CAs aren't very effective at blocking the 'bad' sites. As a result, Let's Encrypt forgoes the pre-issuance checks" - and we know they do because it's all automated - "that CAs have traditionally used to block 'high-risk' [he has in quotes] requests likely to be used for malicious reasons, such as phishing. Instead, Let's Encrypt defers to services like Google's Safe Browsing and Microsoft's SmartScreen, which identify and block dangerous sites at a different layer." Meaning the content layer after the connection has been established and given a green light, a big, happy, oh, look, you got a padlock, you're green, you're secure. Thus the spoofing problem.

He says: "Most of the commercial CAs disagree with Let's Encrypt's position, and this is a topic that is frequently debated. For more background on this topic, I suggest reading a great post from Eric Lawrence, which inspired this post." He says: "I'm not asking Let's Encrypt to change its larger position. I respect and understand its view and think it's a sensible position given its goals as a CA. Given that the content filtering debate is such a heated topic, I would like to sidestep it all together and ask for something much simpler: Stop issuing certificates containing 'PayPal.'" That's it.

"Certificates containing the term 'PayPal' are being pervasively abused, and the continued issuance of these certificates poses a danger to the web by bestowing legitimacy to phishing sites. Let's Encrypt can address this without impacting its users or its mission. That's it. That's all we're asking. Now you may be thinking, wait, isn't this content filtering? As other CAs implement it, filtering involves complicated blacklists, submissions to multiple reputation and spam services, manual review processes, and a constant whack-a-mole game figuring out what misspelling or homonym the phishers are using this week. There is no way for Let's Encrypt to implement similar measures without it compromising its mission or incurring large costs in developing, maintaining, and reviewing such measures." In other words, that's what we pay other CAs to do. And it's why only domain validation certificates are available through Let's Encrypt and not the higher validation certificates.

So he says: "There is no way for Let's Encrypt to implement similar measures without it compromising its mission or incurring large costs in developing, maintaining, and reviewing such measures. I think it is unfair to ask for that. Instead, simply blocking 'PayPal'" - and he has it in quotes, 'PayPal,' that word - "and literally just 'PayPal,' no variations or misspellings, is an easy, feasible, and effective measure against the most dangerous and malicious use of Let's Encrypt certificates. When Eric published his post, Let's Encrypt had issued 709 certificates containing 'PayPal.' Now that number is 988." And that's the end of his post that I quoted.

I jump over to just the beginning paragraph of Eric Lawrence's posting dated January 16, 2017, which he called "Certified Malice." And so Eric writes: "By December 8, 2016" - so early December three months ago - "Let's Encrypt had issued 409 certificates containing 'PayPal' in the hostname. That number is up to 709 as of this morning." So that was from December 8 to January 16.

"Other targets include Bank of America (14 certificates), Apple, Amazon, American Express, Chase Bank, Microsoft, Google, and many other major brands. Let's Encrypt," he writes, "validates only that the, at one point in time, certificate applicant can publish on the target domain. The CA also grudgingly checks with the Safe Browsing service to see if the target domain has already been blocked as malicious, although they 'disagree' that this should be their responsibility. Let's Encrypt's short position paper is worth a read. Many reasonable people agree with it."

And so I just thought this was interesting. Everybody knows I've been promoting and a fan of and thinking that Let's Encrypt is a great idea for years. We've watched it. We've charted it. I've explained its limitations. I don't use it. I use DigiCert. I still, you know, I proudly use them. I think for many places it makes more sense to have somebody creating a higher level of authentication. Let's Encrypt, however, is the benefit of being free and automated. So it gives you privacy. It gives you security. Unfortunately, we're in this problem where the browsers are celebrating the fact that you're over HTTPS, and we've trained users to believe that green is good in the URL, and so secure means something. Unfortunately, it's so spoofable to say www.paypal.com-securityservices.com. Well, the user doesn't understand that the www.paypal.com means nothing. It's the secondary securityservices.com that is actually the domain that the certificate is for. So everybody is going to make this mistake. The only thing I can think we need to do is, from the browser standpoint, start treating domain validation as the unwanted stepchild of TLS connections. Somehow say, well, yes, it's good. You're not in plaintext anymore. But nobody has verified that you are actually at the domain you think you are.

I don't know how we solve the problem. But, I mean, I love that we have a simple way for many sites where it didn't make sense for them to be over TLS, that now they can be. It was completely foreseeable that this system that allowed automated domains to receive certificates would be abused. I do think we have a problem that needs solving.

Leo: Yeah. I have kind of mixed feelings about it, only because...

Steve: Yeah. It is a mixed-feeling problem.

Leo: If you have a certificate from any domain, you can add PayPal to that as subdomain. You know, wildcard domains make it possible. And I understand why the cert people are saying, well, first off, we're offering a free service. We can't afford to do certification validation above and beyond we'll check your site or send you an email. And what they're saying is, well, technically, all the cert says is you have a secure link to a site with this domain name. It's on the user and the browser and other people to figure out if that domain name is a malicious domain name or not. Maybe it is now. Maybe it won't be in the future. Maybe it was, wasn't, but now is. That's just the way of the web.

Steve: Yeah.

Leo: I mean, I agree, they probably should have some obvious, I mean, PayPal's not going to ask Let's Encrypt for a cert, so they should probably have, you know, block Apple, PayPal, Google. That would be an easy thing to do with probably no consequence.

Steve: Yeah. Of course, then, where do you stop?

Leo: Right. That's the problem; right? I mean, and I wonder if even other domain registrars do much beyond validating that, I mean, I know when I got a, what was it, I can't remember who it was, but all they did was send me an email. It's not an extended cert. They don't do any other validation.

Steve: Right, right.

Leo: Well, Steve, we've done some good work here, I believe.

Steve: We have indeed.

Leo: And now it's 10:45, so I think it's time to go.

Steve: And I'm sure we'll do some more good work next week, my friend.

Leo: Every Tuesday, 1:30 Pacific, 4:30 Eastern 21:30 UTC. That's when we convene the Security Now!, it's not a roundtable, it's really more like a kind of square table. Our table, a two-top, for Steve and me to hash it all out. Mostly Steve. I'm just enjoying my tea. You can come here and watch live, if you want, TWiT.tv/live, at that time. You can also watch on YouTube, YouTube.com/twit. We're live there. Many, many, many TWiT apps, five of them on Apple TV alone, all of which stream live. And of course you can get it on demand. Now, Steve hosts an on-demand audio version of the show and a transcript. That's the only place you can get that at GRC.com.

While you're there, pick up SpinRite, the world's best hard drive, recovery, and maintenance utility. Play with all the other tools, do a little ShieldsUP, do a little Perfect Paper Passwords, all of that's there: GRC.com. We store all of the shows at TWiT.tv/sn, as well. And of course subscribe because that's the best way to get it. Then you don't miss an episode. And this is one of the shows - most of the shows have the shelf life of a fish. This show you want to keep listening to. This is one of those where you'll want all the back episodes. So subscribe.

Steve: Shelf life of a sequoia.

Leo: A sequoia. That way you'll never miss an episode. Thanks, Steve. We'll see you next time on Security Now!.

Steve: Thanks, Leo.

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/