Where the Plaintext Is

Description: This week we look at one aspect in which Chrome and Chromium differ, and then at a bit of growth news from the DuckDuckGo folks. Google's Project Zero reports on some terrific detective work, and we look at last week's Patch Tuesday. There's also Microsoft's pending change to the flaws which enabled last year's Zerologon debacle, and the NSA's interesting statement about enterprises and the DoH protocol. We look at the research that cracked the secret key out of Google's supposedly uncrackable Titan FIDO U2F dongle, and we catch up with a bit of listener feedback. Then we wrap up by looking at various aspects of the frenzy caused by WhatsApp's quite predictable move to incorporate its users' conversation metadata into Facebook's monetization ecosystem.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-802.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-802-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Lots to talk about. Our first Patch Tuesday of 2021. Not a record-breaker, only one zero-day. Just one? Just one. We'll also talk about a flaw, a side-channel attack in Google's Titan Security Keys. Uh-oh. And Steve explains why you probably don't have to worry about how secure your Messenger is anyway. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 802, recorded Tuesday, January 19th, 2021: Where the Plaintext Is.

It's time for Security Now!. Oh, I know you've been waiting all week. I know we have a lot of people, oh, can't wait till Tuesday, Security Now!. Here's the guy, Steve Gibson. Yay, the man you've all been waiting for from GRC.com and our security guru.

Steve Gibson: We ought to really consider renaming the podcast What Could Possibly Go Wrong?

Leo: Oh, there's plenty, isn't there, Steve. Oh, man.

Steve: Indeed. So the title of this one came to me, well, I sort of stumbled on it when I was sort of laying out the case that I'll be making at the end of the podcast. So at the moment, the title "Where the Plaintext Is" might not be immediately obvious. But it ends up being a fun title when I end up pulling the pieces together. This is Security Now! #802 for January 19th of 2021. Lots of interesting stuff to talk about. We're going to look at

one aspect in which Chrome and Chromium differ. Because the difference has never really been clear. Turns out Google realized that there was one difference they should have been better enforcing than they were.

Then we're going to take a look at some nice growth news from the unfortunately named DuckDuckGo folks. Google's Project Zero has reported on some terrific detective work they've done and also asked a really important question that we'll take a look at. We're also going to do a little bit of a look back at last week's Patch Tuesday, what lessons there were there, what happened; and also at Microsoft's pending change in the flaws, or the remediation I should say of the flaws, which enabled last year's Zerologon debacle. That's coming with next month's Patch Tuesday, but just want to make sure all of our listeners are ready for it in the IT environment.

Also the NSA's interesting statement about enterprises and, I would say, "and" the DoH protocol, but maybe "versus" the DoH protocol would be putting it better. We're going to look at the research that cracked the secret key out of Google's supposedly uncrackable hardware Titan FIDO U2F dongle.

Leo: Oh, interesting.

Steve: Yeah. And then we catch up with a bit of listener feedback, which allows me to tie up a couple loose ends from our "Out With the Old" podcast last week. Then we're going to wrap up by looking at various aspects of the frenzy caused by WhatsApp's quite predictable move to incorporate its users' conversation metadata into Facebook's monetization ecosystem. And that brings us back to the title of the podcast, "Where the Plaintext Is."

Leo: Oh, boy.

Steve: And of course we do have a fun Picture of the Week. So I think a great podcast for our listeners.

Leo: It wouldn't be Security Now! without a Picture of the Week, would it.

Steve: Thanks to our fabulous listeners who send me a constant trickle of these things, yes.

Leo: All right. Are we ready for the Picture of the Week, Mr. G?

Steve: So this is just fun. Just a cute little cartoon. We have an aviator flying a little one-man jet fighter kind of thing. And he is seen to be saying, "DAMN! Cloud computers!!" because he's flying through a bunch of clouds where they are laced with computers. And of course they're bouncing off of his airplane. One of them says "Bounce!" One says "Pock!" I got a kick out of the fact that one of them is "Ping!" And there on the back it says "Cron!" as it bounced off the machine.

Leo: Oh, that's funny.

Steve: Yeah. So anyway, just sort of fun little cartoon of yes, a different meaning of the term "cloud computers."

Leo: Very funny.

Steve: You don't want to actually have computers in the cloud because...

Leo: No, you don't.

Steve: ...you might fly through them, and then that would be bad. So when is Chrome not Chromium? We all know that Google's worldwide leading Chrome browser is based upon the open source and very solid Chromium core. And this tends to beg the question, how are Chrome and Chromium similar and different from one another? Because as we've said, you've got Safari and Firefox, and then everything else is now Chromium. One of the Chromium-based browsers is Chrome, but there's a whole bunch of others that are all based on the Chromium core.

So at least one way in which Chrome is meant to be different from Chromium is in the way Google-specific features that are available only in Chrome operate. These are things like Chrome sync and Click to Call, or signing into the user's Google account and transacting personal sync data - bookmarks, passwords, history, open tabs, settings, preferences, and in some cases even payment info saved in Google Pay. That's meant to be Chrome-specific.

Well, it turns out the Chrome API provides direct hooks - that is, the Chrome API as opposed to Chromium API. The Chrome API provides direct hooks into those services that no other third-party web browser is meant to have. Yet Google Chrome Engineering Director Jochen Eisinger explained that during a recent audit they discovered that some of the third-party Chromium-based browsers were integrating features - whoops - that were only intended for Google's use with Chrome. These APIs were intended to be private and specific only to Google Chrome. But apparently there was no enforcement of that. They were never intended to be integrated into any non-Google, non-Chrome browser use.

He declined to indicate which browsers had integrated Chrome sync without authorization, but he did say that Google would be blocking this unintended behavior starting on the 15th of March, so two months from a couple days ago. By removing access to Chrome sync for other Chromium web browsers, it will remove their ability, which apparently they've had kind of quietly, to integrate the Chrome sync API to sync their users' data to all devices where they're logged into their Google account. However, Google did explain that users who have accessed private Google features such as Chrome sync while using third-party browsers will still be able to access this synced data locally or in their Google account, depending upon their sync settings. They'll also be able to manage their data by going to the My Google Activity page, as well as downloading it from the Google Takeout page, and/or by managing it, also delete it by going there.

So anyway, I thought this was sort of interesting, that Google is very generously sharing the Chromium engine with all other browsers. It doesn't seem to be hurting Chrome's market share any. But there was some stuff that was like, no, no, that's just for us, which they weren't enforcing before. So they're now going to add some enforcement to that, so that it only runs in their own browser. And again, not a big loss for anybody, just sort of one fewer feature that they shouldn't have had. But you can still get it if you just go to the Google pages themselves.

Though it's not a browser, this bit of news is related to our browsing. Not long ago we talked about the privacy-centric DuckDuckGo search engine, and I had some fun talking about, well, actually, Leo, you were clueing me in on some children's game after which this search engine was unfortunately named. And we imagined that, much as "google it" has obtained a generally understood meaning, that perhaps someday the same will happen with "duck it." Although I'm not holding my breath on that one.

Leo: Yeah, maybe not, yeah.

Steve: Yeah, I don't know.

Leo: Is that what they're pushing?

Steve: I'm hearing some people saying, I saw someone said "duckduck," and I don't think that works, either.

Leo: No.

Steve: So, you know, this is the cross this poor guy's going to have to bear in deciding to name his search engine DuckDuckGo. I mean, okay. Maybe, since Google was kind of whimsical, they could get away with it. But no. Anyway, despite its name, the good news is it is actually a serious contender in the rather rarefied search engine space these days. So I wanted to take a moment to share a milestone that it reached for the first time last Monday. Last Monday for the first time in its 12-year history, DuckDuckGo recorded its first day ever of more than 100 million user search queries. For the past two years, DDG - which is a less painful abbreviation - has been in a period of sustained growth, and especially since last August, when they began seeing more than two billion search queries a month on a regular basis.

Okay. Now, to give that some context, however, since that seems like a lot, right, two billion search queries a month regularly, Google does five billion a day. So, okay, they're not at Google's level, obviously, yet. But they're growing. I put a chart in the show notes which is interesting. This shows it anchors on last Monday, well, actually Monday before last, but it shows the 14-day region around there, starting with January 1st, where they were at 77.1 million queries. And then they get up into the mid- to high 80s. And then on the Sunday before that Monday, on the 10th of January, they were at 94.8 million. And then Monday they went to 102. They've since, on the Tuesday-Wednesday-Thursday, they didn't quite make 100 million, but they were in the very high 90s - 99, 98, 97.

So anyway, they're there. And clearly the idea of searching without dragging all of Google's tracking and advertising baggage along is appealing to people. They have also, I wanted to note, expanded beyond their own browser URL as the only way to get there. They've got mobile apps for Android and iOS, as well as a dedicated Chrome extension for DuckDuckGo browsing. And in a tweet last September they indicated that the apps and extension have been installed more than four million times.

Oh, and in another win for the company, DuckDuckGo has been selected as the default search engine in the Tor browser. Remember once upon a time that was Firefox. Now DuckDuckGo. And it will often appear as the default search engine in the private browsing modes of several other browsers, which is kind of nice. You switch into private browsing, and you get a privacy-oriented search engine as part of that. So anyway, just

to sort of keep it on our listeners' radar. I wouldn't be at all surprised if our privacy and security-conscious users are counted among many of DuckDuckGo's users. Despite the name.

Last Tuesday Google's Project Zero published a six-part series which takes a very deep analytical look at a discovery the group made a year ago, that is, during the first quarter of last year. Last week I was talking about watering hole attacks where a browser vulnerability, in this case it was that SCTP protocol vulnerability, where unlike a JavaScript vulnerability which is occurring locally using code in your browser, in this case in this SCTP protocol, in order to be exploited, the user would have needed to visit, often being lured to a so-called "watering hole" server. So I especially appreciated the way Google framed this campaign of theirs. Here's how they began the first of these six pages in this very substantial posting.

They said: "At Project Zero we often refer to our goal simply as 'make zero-day hard.'" This is not the first time we've talked about this. I think it's exactly the right goal. They said: "Members of the team approach this challenge mainly through the lens of offensive security research" - as opposed to just responding to, how do they prevent. They said: "And while we experiment a lot with new targets and methodologies in order to remain at the forefront of the field, it is important that the team doesn't stray too far from the current state of the art. One of our efforts in this regard is the tracking of publicly known cases of zero-day vulnerabilities." In other words, they look, really look at each of them and ask a bunch of questions about them.

They said: "We use this information to guide the research. Unfortunately, public zero-day reports rarely include captured exploits, which could provide invaluable insight into exploitation techniques and design decisions made by real-world attackers." They said: "In addition, we believe that there is a gap in the security community's ability to detect zero-day exploits. Therefore, Project Zero has recently launched our own initiative aimed at researching new ways to detect zero-day exploits in the wild. Through partnering with the Google Threat Analysis Group (TAG), one of the first results of this initiative was the discovery of a watering hole attack in the first quarter of 2020, performed by a highly sophisticated actor."

They said: "We discovered two exploit servers delivering different exploit chains via watering hole attacks. One server targeted Windows users, the other targeted Android. Both the Windows and the Android servers used Chrome exploits for the initial remote code execution. The exploits for Chrome and Windows included zero-days. For Android, the exploit chains used publicly known n-day exploits." And of course referred to as "n-day" because, as I've often noted, they're only true zero-days if they are not previously known. So if it's a publicly known thing, then by definition it's an n-day. Then they said: "Based on the actor's sophistication, we think it's likely that they had access to Android zero-days, but we did not discover any in our analysis."

So I have a picture in the show notes of the chart that they provided along with this analysis, where they show on the left incoming, a number of affected websites - 1, 2, ... to N. And those websites deliver a pair of iframes which have no contact or affiliation with each other, other than that they are sourced from the same set of servers. And then one iframe will pull content from the Windows exploit server, because remember an iframe itself is a reference to some third-party server that it pulls content from to fill the frame. The other iframe pulled its content from the Android exploit server. So both of them would do that. And presumably they only pull content based on the platform they find themselves running in.

So if the Windows content-pulling iframe sees it as running on a Chrome browser on Windows, then it pulls the content. And whereas the Android iframe would just leave itself inert and pull nothing. Whereas if it finds that it's running Chrome on Android, then

the Windows iframe would not bother pulling iframe exploits into its frame, whereas the Android iframe would. So from that point, then, the Windows exploit iframe performs some Chrome renderer exploits, and the Android exploit, if it's on Android, performs one of a number of Chrome renderer exploits, those being the problems that Google discovered being exploited by this particular attack.

So, and of course, one of the things you note is that, since the multiple servers apparently are serving this content, this could well have been a malvertising campaign since, as we know, ads run in iframes on their hosting pages.

Anyway, so they continue their explanation, saying: "From the exploit servers, we have extracted" - so the beauty of this is that they discovered this campaign while it was underway, which is what differentiates it from the typical zero-days they learn about from third parties. They found this one, which allowed them to extract a great deal of knowledge about the attackers because it was still alive. So they said: "From the exploit servers, we have extracted renderer exploits for four bugs in Chrome, one of which was still a zero-day at the time of the discovery," meaning that they found something they didn't know about, a bug they didn't know Chrome had when they saw this thing working.

They also found two sandbox escape exploits abusing three zero-day vulnerabilities in Windows, and a privilege escalation kit composed of publicly known n-day exploits for older versions of Android. And of course we know, unfortunately, older versions of Android are really much a large population of Android. So even though they're publicly known, they're still very likely to be active. They said: "The four zero-days discovered in these chains have been fixed by the appropriate vendors."

And they said: "We understand this attacker to be operating a complex targeting infrastructure, though it didn't seem to be used every time. In some cases, the attackers used an initial renderer exploit to develop detailed fingerprints of the users from inside the sandbox. In these cases, the attacker took a slower approach, sending back dozens of parameters from the end users device before deciding whether or not to continue with further exploitation and use a sandbox escape. In other cases, the attacker would choose to fully exploit a system straight away, or not attempt any exploit at all. In the time," they said, "we had available before the servers were taken down, we were unable to determine what parameters determined the 'fast' or 'slow' exploitation paths."

"The Project Zero team," they said, "came together and spent many months analyzing in detail each part of the collected chains. What did we learn? These exploit chains are designed for efficiency and flexibility through their modularity. They are well-engineered, complex code with a variety of novel exploitation methods, mature logging, sophisticated and calculated post-exploitation techniques, and high volumes of anti-analysis and targeting checks." So these exploits were themselves operating very defensively.

They said: "We believe that teams of experts have designed and developed these exploit chains. We hope this blog post series provides others with an in-depth look at exploitation from a real-world, mature, and presumably well-resourced actor. The remaining posts in this series share the technical details of different portions of the exploit chain, largely focused on what our team found most interesting."

They said: "We include detailed analysis of the vulnerabilities being exploited and each of the different exploit techniques; a deep look into the bug class of one of the Chrome exploits; and an in-depth teardown of the Android post-exploitation code." They said: "In addition, we are posting root cause analysis for each of the four zero-days discovered as a part of these exploit chains. Exploitation aside, the modularity of payloads, interchangeable exploitation chains, logging, targeting, and maturity of this actor's operation set these apart. We hope that by sharing this information publicly, we are

continuing to close the knowledge gap between private exploitation, what well-resourced exploitation teams are doing in the real world, and what is publicly known."

And again, this is chilling, a bit like the whole SolarWinds revelations were. The idea that there is this kind of high-level, sophisticated, methodical, careful, and clearly in many cases very targeted use of currently unknown vulnerabilities sort of should be sobering for us because the idea is that these actors know whose systems they want to get into, whose Android phones they want to penetrate, whose Windows systems they want to penetrate. They arrange to get them to visit a server somehow hosting some content which then puts these iframe payloads onto their browser window. We know that the chances are very high that an Android user will be using Chrome, a Windows user will be using Chrome, and they found ways to break through, you know, these renderer exploits, as in page rendering exploits. So they have rendering flaws that let them escape from Chrome's deliberate containment, and then they go about doing what they want to do.

For anyone who's interested in following up, I do have a link to this Google Project Zero blog page, which is the first of this multipage presentation, for anyone who's interested. But what I thought to be the single most important phrase in what I read was where the Project Zero guys wrote: "In addition, we are posting root cause analysis for each of the four zero-days discovered as part of these exploit chains." The key words there were "root cause analysis." In other words, "Oh, crap. We just found another zero-day." Yeah. Always definitely better to have found it than not. But much better for it never to have existed in the first place.

So, rather than just fixing it and saying "Whew" and then going to lunch, let's instead order in and spend lunchtime asking ourselves how this fault was introduced in the first place, and what takeaway lessons we can learn to preemptively prevent anything like this from happening again. That is essentially what they're doing. They are looking at every one of these zero-days carefully, the code that surrounds it. How did it happen? And I think this is exactly what needs to be done. And clearly the continuing prevalence of software bugs which beleaguer our industry suggests that examining root causes occurs far too infrequently.

I've commented here on the podcast in the past that I truly love solving problems and puzzles with code. For me it's a passion and a craft. So whenever I find a bug in my own code, as I've said before, I come to a dead stop, and I spend some time asking myself and exploring how that bug happened in the first place. What was I thinking? Exactly what did I get wrong? Is it likely to have happened elsewhere? Do I have a bad habit that needs correcting? In other words, rather than being embarrassed about a bug, be informed by it. Some incorrect way of thinking caused it.

And so I would submit that the only way for any craftsman to improve at their craft is to rather carefully examine and understand their mistakes. Certainly, Leo, we know this is the case with chess; right? The reason all of the really high-end masters record every move of their game is they're going to take a look at that later and spend a lot of time examining the board at each stage and understand why they lost the game.

Leo: You bet, yeah.

Steve: If in fact they did. At a certain point it's the only way. You're just not going to get better by hoping to do better next time. At some point you really have to understand the nature of the mistakes you're making in order not to make them again. And there have been instances where I have found a bug where I realized, oh, my god, there are others of these. And I've designed...

Leo: That's a bad feeling.

Steve: ...designed a regular expression, a regex, which will find the other places where I almost certainly did the same thing wrong, and fixed other ones.

Leo: It's hard for people because we're blind to our own mistakes often. It's so hard to do your own debugging because you stare and you stare and you stare at the code, and you go...

Steve: Looks great.

Leo: Looks great. I don't understand. I can't, you know. But I agree with you. There is a certain sport to it, as well. It's kind of fun.

Steve: Oh, it is. And when you are stepping through the code, and the debugger shows you, okay, I'm moving this into EAX, and it's like, wait a minute, I just wiped out what was there, which I'm going to be needing two steps later.

Leo: It must be so easy to do that in assembly code because any modern high-level language has all sorts of protections so that you avoid that kind of thing. But it's all on you in assembler.

Steve: Well, it is. But what I like about it is nothing is hidden. There are no, like, ooh, I meant for this to be unsigned, and now blah blah blah. It's like, no, there's no such thing in assembler. So, yeah. We have the first Patch Tuesday of the year 2021. Okay. So we didn't break any records.

Leo: Aw.

Steve: And that's the good news. We had a total of 83 vulnerabilities. And we were like over a hundred several of the early months last year. Ten of those 83 were Microsoft classified as critical, and another was a zero-day, which we'll get to in a moment.

Leo: Ooh, not good.

Steve: Yeah. Not good. But good that it's patched, and hopefully everybody's updated their machines. I just got the little notice while you were talking, Leo, that this machine, this Windows 10 machine that I'm Skyping on will be restarting, but not during the podcast, fortunately. Two of the critical vulnerabilities that were fixed were remote code execution vulnerabilities in two codecs. One was a Microsoft DTV-DVD video decoder, and the other was the HEVC video extensions, which just had a problem recently. So yes, yet again.

There was also a remote code execution vulnerability in the GDI+ library, the Graphics Device Interface, and a critical memory corruption vulnerability in Microsoft Edge, in the browser. The remaining five critical remote code execution vulnerabilities were all found

and fixed in Microsoft's RPC, the Remote Procedure Call runtime, which is really not where you want to have remote code execution vulnerabilities because RPC is inherently exposed on the network. The biggie was the zero-day that was found in Microsoft Defender, that is, in the AV system, being executed in the wild. They had found a mistake in Defender. And when the malware was being scanned by Defender, it took it over. So, whoops.

Leo: Oh. Oh. Oh.

Steve: It's being tracked as CVE-2021-1647, a remote code execution flaw in Microsoft's Malware Protection Engine component, mpengine.dll. Sometimes when I look at Task Manager and browse through what the heck is going on, I'll see, oh, yeah, mpengine.dll. So the good news is mine on Windows 7, I don't even know if it's good or bad, come to think of it, but let's hope not. Most of them now are on Windows 10. Oh, and a proof-of-concept exploit for the flaw is public. So we don't have any indication of how widespread the exploitation is or was, nor anything about its nature. But we do know that it was discovered taking over instances of Defender. So by now we are post-patch by a week. I'd imagine that everyone has updated and rebooted. If not, I don't think this is like a pants-on-fire problem. But at least know that there is a bad problem that was a zero-day that was fixed. So good to reboot when you can.

Also, with next month's Patch Tuesday, which will occur on February 9th, we have the final second phase of Microsoft's Zerologon remediation. As we know from its extensive abuse last year after its discovery and patching, which did very little to prevent it from being exploited post-patch. Thich demonstrates how little patching is, amazingly, going on. This was the Zerologon flaw in Microsoft's previously believed to be more secure than it turned out to be RPC Netlogon protocol, which provided the perfect means for attackers to move laterally within an organization after they had once established a foothold somewhere. It allowed them to effortlessly log onto the enterprise's Master Domain Controller, which then literally gave them the keys to the kingdom.

So the first of the two patches was in August, which fixed the security problem between Windows devices to reinforce truly secure Remote Procedure Call communications for machine accounts on Windows devices, trust accounts, as well as all Windows and non-Windows Domain Controllers. In other words, as of August, Windows to Windows RPC was secured. But there's a large population of non-Windows devices that are also users of RPC. They were not previously understood to be as insecure as we came to understand they were because RPC, the Netlogon protocol, turned out you could put some zeroes in some fields, and everything would be just fine.

So since August, IT admins have been able to log all non-secure use of RPC by various devices within their infrastructure in order to help them prepare for what's going to happen on February, which is drop-dead day for all non-Windows devices, which have had this period, sort of a grace period, during which they could get themselves updated to be using secure RPC as have all Windows systems since August. After February 9th, Domain Controller enforcement mode, as it's called, will be enabled by default. Domain Controller enforcement mode requires that all Windows and non-Windows devices use secure RPC with Netlogon secure channel, unless customers explicitly override that. And, you know, good luck to you, if you choose to.

You can force it off if you've got some non-compliant device still in your system that you have to still be able to support. But at that point, Microsoft is going to just wash their hands of you and say, okay, this is not our fault, if you now get ransomware that uses the fact that you've allowed non-secure channel Netlogon because you have some thing that you just have to still keep using that refused to be updated, some legacy box of

some kind, and its publisher is gone, but you have to keep using it. Who knows what the scenario would be. But hopefully that's not the case. Next Patch Tuesday the final bit of deliberate insecurity of the Netlogon protocol will be eliminated. So hopefully everybody will be able to step up at that point.

Now, in an interesting posting from the NSA, they are - the NSA, the National Security Agency are warning against outsourcing of DoH services. In their seven-page advisory titled "Adopting Encrypted DNS in Enterprise Environments," which they posted last Thursday, the NSA warned against outsourcing DoH services to a third-party provider. Which at first glance might seem a little odd. But, okay. So here's their executive summary. The first bit of it is, yeah, a little obvious to all of us.

They said: "Use of Internet relies on translating domain names like 'nsa.gov' to Internet Protocol addresses." Right. "This is the job of the Domain Name System (DNS). In the past, DNS lookups were generally unencrypted, since they have to be handled by the network to direct traffic to the right locations. DNS over Hypertext Transfer Protocol over Transport Layer Security (HTTPS), often referred to as DNS-over-HTTPS (DoH), encrypts DNS requests by using HTTPS to provide privacy, integrity, and 'last mile' source authentication with a client's DNS resolver." All that's true, in case any of you haven't been paying attention in the last couple years.

Then they said: "It is useful to prevent eavesdropping and manipulation of DNS traffic. While DoH can help protect the privacy of DNS requests and the integrity of responses, enterprises that use DoH will lose some of the control needed to govern DNS usage within their networks unless they allow only their chosen DoH resolver to be used. Enterprise DNS controls can prevent numerous threat techniques used by cyber threat actors for initial access, command and control, and exfiltration."

So they said: "Using DoH with external resolvers can be good for home or mobile users and networks that do not use DNS security controls. For enterprise networks, however, NSA recommends using only designated enterprise DNS resolvers in order to properly leverage essential enterprise cybersecurity defenses, facilitate access to local network resources, and protect internal network information. The enterprise DNS resolver may be either an enterprise-operated DNS server or an externally hosted service. Either way, the enterprise resolver should support encrypted DNS requests, such as DoH, for local privacy and integrity protections, but all other encrypted DNS resolvers should be disabled and blocked.

"However," they said, "if the enterprise DNS resolver does not support DoH, the enterprise DNS resolver should still be used, and all encrypted DNS should be disabled and blocked until encrypted DNS capabilities can be fully integrated into the enterprise DNS infrastructure. This guidance explains the purpose behind the DoH design and the importance of configuring enterprise networks appropriately to add benefits to, but not hinder, their DNS security controls. The following recommendations will assist enterprise network owners and administrators to balance DNS privacy and governance." And I will blessedly spare all of us from any further of that.

But for what it's worth, I completely agree with every sentence in that Executive Summary. I think they got it all exactly right. The only issue I might take is wondering about the value, let alone the necessity of enabling DoH within the enterprise at all. I really don't see what value it provides if your internal LAN has encrypted DNS or not. Maybe that's a function of how big the encrypted LAN is. I mean, for example, once upon a time HP was, what, it was 15-dot. I think 14-dot and 15-dot. And if they somehow glued all of that into one single massive enterprise LAN, then okay, maybe it would make sense to perform some encryption, just because you've sort of created at that point kind of a quasi public LAN, it's so big.

But I really take their point. I think what they're responding to is that there are a lot of individuals inside of the enterprise that are saying, hey, cool, I can blind my enterprise's IT to everything I'm doing. You know, tech and privacy-minded employees might want to sneak their browser traffic out of the organization without being monitored. But we always remind everyone that the use of employer networks is the employer's to oversee and regulate. So I would think that enterprises would be entirely correct to block the use of DoH and require browsers to use standard DNS and the organization's DNS with whatever added security filtering and malware detection the enterprise might wish to deploy. And yeah, bring up DoH in the enterprise and then users' browsers can use that.

But anyway, I appreciated, I guess, the NSA just kind of pointing out that circumventing all of an enterprise's controls by immediately tunneling out to an external provider is probably not the right thing to do. So anyway, this posting of the NSA was picked up by all of the various security monitoring websites. So I just thought I'd take a moment to say, yeah, I think I see their point completely.

A side channel in Titan. The guys at NinjaLab performed a classic side-channel attack on Google's Titan FIDO U2F security key. By watching it work, while monitoring its electromagnetic RF radiation, they successfully extracted its embedded, super-secret, deliberately designed to never be extracted, private key. Yeah.

Leo: Is it the same private key in all the Titan keys?

Steve: No. Each key had its own.

Leo: No. Each key has its own, of course, yes.

Steve: Yes. But the whole point is all of the crypto is done on key.

Leo: Right.

Steve: So it never exposes that private key. And once you have it, then you are 100% able to spoof the presence of that key. That's the whole point.

Leo: Do you have to have physical access to the key?

Steve: Yeah.

Leo: Okay.

Steve: So I'll explain all this in a second. We've talked about side-channel attacks a lot in the past. Sometimes the power drawn by a device while it's performing cryptographic operations will fluctuate a little bit. Sometimes the power supply, we've talked about this, on a computer might emit different sounds. Or perhaps the timing required to perform an operation will vary. It could be anything. The coolest way of stating the goal of avoiding any possibility for a side-channel is that nothing about the behavior of a device should vary as a function of any internal secret bits.

So taking timing as an example, the time required to perform some operation, typically the pattern of conditional jumps taken in the code, should be completely independent of the secret being kept. That is, you should never have a jump taken or not based on bits of a secret key. That would be just really bad. So the bits of the secret key must not cause the algorithm's code path to vary, as one of the base requirements for avoiding side-channel attacks. The fact is, some things are feasible to hold constant; but others I would argue, much as I argued against Intel's claim of having a ransomware detector in their processor last week, I would argue that some things really do fall below one's ability to control. Such as the instantaneous electromagnetic radio frequency radiation of an integrated circuit that's doing the work.

Leo: Yeah. You can't really - I guess you could wrap it in tinfoil. I don't know.

Steve: Yeah. And the problem is, then you unwrap it. I mean, even if you put like a lid on it, well, you pop the lid, and there it is. So the NinjaLab guy said: "The Google Titan security key is a FIDO U2F hardware device proposed by Google, available since July 2018, as a two-factor authentication token to sign into applications - for example, your Google account." They said: "Our work describes a side-channel attack that targets the Google Titan Security Key's secure element" - and that's the NXP, which used to be Philips, they renamed, NXP A700x chip - "by the observation of its local electromagnetic radiations during ECDSA (Elliptic Curve DSA) signatures, the core cryptographic operation of the FIDO U2F protocol. In other words, an attacker can create a clone of a legitimate Google Titan Security Key." Period.

"To understand the NXP Elliptic Curve DSA implementation, find a vulnerability and design a key-recovery attack, we had to make a quick stop on Rhea (R-H-E-A)." That's the NXP J3D081 Java Card smartcard, they said, that is freely available on the web. "This product looks very much like the NXP A700X chip and uses the same cryptographic library. Rhea, as an open Java Card platform, gives us more control to study the Elliptic Curve DSA engine."

So that was very clever. They realized that these two chips, the one they knew nothing about and the one they could know everything about, were very similar. So they trained themselves up using the open source full everything is known about it version, and then cross-applied that to the one they didn't know anything about.

They said: "We could then show that the electromagnetic side-channel signal bears partial information about the Elliptic Curve DSA ephemeral key. The sensitive information is recovered with a non-supervised machine learning method and plugged into a customized lattice-based attack scheme." Okay. So they said: "Finally, 4,000 ECDSA observations were enough to recover the known secret key on Rhea and validate our attack process. It was then applied on the Google Titan Security Key with success, this time by using 6,000 observations, as we were able to extract the long-term Elliptic Curve DSA private key linked to a FIDO U2F account created for the experiment." In other words, the golden goose, the keys to the kingdom.

Then they said, as a cautionary note, they said: "Two-factor authentication tokens like FIDO U2F hardware devices' primary goal is to fight phishing attacks. Our attack requires physical access to the Google Titan Security Key, expensive equipment, custom software, and technical skills. Thus, as far as our study goes, it is still safer to use your Google Titan Security Key or other impacted products as FIDO U2F two-factor authentication token to sign into applications rather than not using one." And by the way, the Yubico uses the same Philips chip. So it was conjectured that it would be subject to the same side-channel attack.

They said: "Nevertheless, this shows that the Google Titan Security Key and other impacted products would not avoid unnoticed security breach by attackers willing to put enough effort into it. Users that face such a threat should probably switch to other FIDO U2F hardware security keys, where no vulnerability has yet been discovered." And I'm thinking, well, okay, but it hasn't been discovered until it has been. So the point is, don't let bad guys take your Google Titan Security Key.

Leo: And if you do, if you lose control of it, just deauthorize it, and then they can't do anything with it. Every single dongle-related site that I use my YubiKey on has the ability to remove a YubiKey and add a new YubiKey. I've done it many times. I mean, the saving grace on this is they have to have access to the key.

Steve: Right, right. And some secrets cannot be kept. And the fancier the system is that tries, the more likely it is to have some behavior that gives it away. Unlike a time-based one-time token, which uses symmetric cryptography, where each end needs to share a private key, systems like FIDO and also my own SQRL, use asymmetric public key cryptography where the user holds their private key, and the remote end holds their public key. This significantly reduces the risk to the user and to the system, since the public keys never need to be kept secret. The takeaway for this hardware token hack is not to be too glib about the idea that the hardware token's private key cannot possibly be extracted. It clearly can be, if someone is sufficiently determined and, as you said, Leo, if you accept the challenge of some hacker who says, oh, let me borrow your...

Leo: You don't need your key; do you?

Steve: Yeah, don't let him have your key.

Leo: You wouldn't give him your house key. Don't give him your Titan key, either.

Steve: That would be a bad idea, yeah.

Leo: Just borrow your house key for five seconds.

Steve: Speaking of tokens, I got an interesting tweet from Robert Rosenberg which I got a kick out of. He said: "Several podcasts ago, you were discussing the PayPal football." He says: "Well, I did it. I took it apart." He says, in parens: "(You've got to be careful, but it's not hard.)" He says: "Found the battery type, bought one, and put it in. Next time I dealt with PayPal, I used the now-working football, and PayPal took it. They authenticated on my dead-for-years token." And he says: "I'm guessing it's a press-based one-time password, not a time-based one-time password."

And I completely agree with Robert. I think that we must have known that once, but I forgot, since I do recall, as I've mentioned before, noting that our listeners discovered that one of the digits was a simple modulus 10 counter. So it must have been that they waited for it to turn off, then pressed it again and noted that one of the digits was simply incrementing 0 through 9 and back again. So it isn't clock-based. There wasn't a timer in there. That really makes a lot more sense, just in terms of all the problems that you would have with it becoming out of time sync. A smartphone doesn't have that problem

because it's on the cellular network and is getting a constant update of its clock. So anyway, just very cool, Robert. Thank you for sharing. If anyone's interested, if anyone still has the football, you can change the battery and bring it back to life.

Leo: Good to know.

Steve: Very, very cool. Peter Sinclair said, regarding "Out With the Old": "Great show." He says: "Steve took me right back to the '80s. I agree with you stripping out the old code from SpinRite 6.1. We still have our 6.0 or older copies if we want to run it on any heritage drives we might have lying around." And I wanted to mention I'm so glad that Peter noted that, since I had forgotten to explicitly mention last week that all owners of SpinRite will continue to have access to previous versions 5 and 6.

Just yesterday, I removed support for diskette drives from the 6.1 code. Again, since they are so different from all other mass storage drives, and there was a bunch of explicit special case handling support for diskettes laced throughout the code, which is almost doing nobody any good. But diskettes are a medium that can benefit significantly from SpinRite. And every so often someone, typically an archeologist, will come up a diskette that's no longer readable, but that still contains some important data. In such cases we suggest they download SpinRite 5, which all SpinRite 6 owners are able to do, because oddly, for some reason, v5 appears to do a better job on diskettes than v6.

And believe me, this has and continues to bug me. I've spent hours staring at the code, and I can't explain why that's the case. But I very clearly remember testing this, and SpinRite 5 was better than SpinRite 6 on diskettes. So anyway, since 6.1 won't even see the diskette, you will be easily reminded to go get an earlier version of SpinRite. And they cohabitate on your SpinRite boot USB or CD or whatever. So it's easy to have them all around. Anyway, just wanted to thank Peter for allowing me to remind our listeners that, as we move forward, it's not like the old versions die or don't work anymore or can't be used any longer. They do still exist.

Leo: Do you offer them for download, though?

Steve: Yeah.

Leo: You can go there? Nice.

Steve: You're actually to actually - yeah, you're able to actually, in the download link, just change it to SpinRite 5, and you get that version of SpinRite.

Leo: That version, yeah. That's really good. That's cool.

Steve: Yeah. And finally Oaksong, tweeting as @oaksong, he said: "@SGgrc, glad to know I can still use those OnTrack managed drives." And we know he's joking, or hope he's joking. But here again, SpinRite 6.1 won't have any support for that. It'll just look at, I mean, it'll see the drive and run, but it won't see the wacky partitions that it contains, whereas SpinRite 6 will. So everybody will be able to use SpinRite 6, if you do run across an OnTrack managed drive.

Leo: Who knows; you know? I bet there's some out there. Maybe even storing some bitcoin wallets.

Steve: Given our listeners, Leo, I guarantee it.

Leo: Saw the guy who's offering $20 million to some town, I think it's in New Jersey, if they will find his discarded hard drive with hundreds of millions of dollars of bitcoin on it.

Steve: Ohhh.

Leo: And the other one, the guy who put it on one of those locked hard drives, he only has two more...

Steve: Right, he's got two guesses left.

Leo: He says: "I've come to grips with it."

Steve: Ohhh.

Leo: Ohhh. His is worth 240 million. Yikes. That's a life-changing amount of money, I think. "Where the plaintext is."

Steve: So WhatsApp has shown us that it's not only Zoom's CEO Eric Yuan who is deftly able to somehow locate the exact center of a large pile of poo to step in.

Leo: Nicely put.

Steve: In Eric's case, well-meaning though he was, every time he opened his mouth in a well-meaning attempt to clarify something, he made things worse, often much worse. And now, over the past week, in an apparent gross misunderstanding of why their own users were using their secure messaging product in the first place, WhatsApp, as we know, has managed to trigger such an exodus from their platform that the two primary alternatives, Telegram and Signal, have never seen such increases in their user base.

And in fact Signal has continued having trouble staying online amid the deluge. We reported last week that Signal was struggling to authenticate the torrent of incoming users since they require an SMS messaging loop, and the cellular carriers were apparently having scaling issues of their own. Then that trouble apparently migrated from the authentication side to the service itself.

For example, this triggered Signal outages and a tweet last Friday from Edward Snowden. Edward tweeted: "For those wondering about @SignalApp's scaling, #WhatsApp's decision to sell out its users to @Facebook has led to what is probably the biggest digital migration to a more secure messenger we've ever seen. Hang in there while the Signal team catches up." And then signed Edward Snowden, January 15, 2021.

And the rush may have been further accelerated when Elon Musk simply tweeted the two words "Use Signal" to his 42.6 million followers.

Leo: It's so powerful. It's amazing. Just amazing.

Steve: So one of the questions that inevitably arises from any use of a free service is why is it free, and who is paying for this? This, of course, reminds us of the famous joke: "Well, yeah, we do lose money on each one. But don't worry, we make it up in volume." So this is one of the reasons why, if I were to use any third-party messaging app, and I don't because I just don't have any need for any, I would still choose Threema, which is paid for one time, although there are some enterprise subscription deals. And that one time generates operating revenue for the company. Oh, yeah, and as I mentioned, their so-called "Threema Work" product for the enterprise has a subscription model. So again, we understand how they are monetizing.

And just as a reminder, since I mentioned Telegram and Signal, I just checked in with Wikipedia to see what they said about Threema. They said: "Threema was founded in December 2012 by Manuel Kasper. The company was initially called Kasper Systems GmbH, German. Martin Blatter and Silvan Engeler were later recruited to develop an Android application that was released in 2013. In summer 2013, the Snowden leaks helps create an interest in Threema, boosting the user numbers to the hundreds of thousands. When Facebook took over WhatsApp in February 2014, Threema got 200,000 new users, doubling its user base in 24 hours. Around 80%, eight zero percent, of those new users came from Germany, which I guess makes sense since it's a German company. By March 2014, Threema had 1.2 million users. In spring 2014, operations had been transferred to the newly created Threema GmbH.

"In December 2014, Apple listed Threema as the most sold app of 2014 in the German App Store. In 2020, last year, Threema expanded with video calls" - we talked about that at the time - "plans to make its codebase fully open source, as well as introduce reproducible builds and Threema Education, a variation of Threema intended for education institutions." And, finally, Wikipedia is up to date. "During the second week of 2021" - which is to say, last week - "Threema saw a quadrupling of daily downloads spurred on by controversial privacy changes in the WhatsApp messaging service. A spokesperson for the company also confirmed that Threema had risen to the top of the charts for paid applications in Germany, Switzerland, and Austria. This trend continued into the third week of the year with the head of marketing and sales confirming that downloads had increased to 10 times the regular amount, leading to 'hundreds of thousands of new users each day.'"

So, yeah. If I had any need for private messaging, Threema would be my choice. I would carefully and manually exchange my Threema public key with the people I need to converse with secretly, and that would be that. When I was working with Rasmus on the SQRL XenForo integration, at his suggestion we used Signal, not because we needed any super secrecy, but because it had a convenient Windows desktop client, and we were in different time zones. So we were able to easily exchange messages, screenshots, and links that way. And when we happened to both be awake at the same time, our interaction would go interactive. And it was comforting to know that our exchanges could not be intercepted and decrypted. But these days that's sort of expected; right?

But I'm interested in the massive effect that was driven by WhatsApp's statement that they intend to share messaging metadata with Facebook's other businesses. It's as if everyone using WhatsApp believed that Facebook purchased WhatsApp out of the goodness of their heart in order to offer the entire world a free messaging service without any strings. As we noted above about the influx to Threema when Facebook announced

their purchase of WhatsApp in the first place, even back then not everyone believed in the free lunch theory.

So when Facebook dropped their other shoe about their plans for further monetizing their WhatsApp users, no one should have been surprised. It's always been clear that sooner or later Facebook would do this. And if they cannot peer into the conversations, then they'll at least collect everything they're able to from the outside. It seems to me that knowing who I'm talking to and when, and for how long, and where we're located, is an intrusion. But hey, it's free.

The problem was that they made, I think, too large a change all at once. It's like the old joke about how to boil a frog. You don't toss a frog into boiling water because it will jump right out. Instead, you place the frog into cold and comfortable water, then slowly turn up the heat. In retrospect, Facebook should have patiently and very slowly chipped away, little by little, at the privacy of their WhatsApp users through a series of much smaller privacy incursions. There is a very non-zero switching cost associated with changing messaging platforms. Right now Twitter is full of people complaining that they just switched to Signal, but no one they know is there. So it's not enough just to make the move, you need to move your entire community. And that's often difficult in the extreme.

So if Facebook had instead chipped away at their users' privacy, at each step the user would reconsider the switching cost of leaving WhatsApp, where all of their friends already are, and they might recall their disappointment the last time they tried. Versus the loss of just one additional little privacy aspect that really doesn't matter that much anyway; right? And if that had been the strategy, it seems clear that many fewer would have left. Now, of course, they've said, oh, wait a minute, we're going to postpone for 90 days, and everybody misunderstood what it was we were doing. We're going to try to be much better about communicating this. Right.

And of course, as we know, instead of understanding that this incremental approach should have been taken, WhatsApp and Facebook, it sort of seems like maybe they got a little too big for their britches and lowered the boom on their users all at once, as we talked about last week, the "agree to this or else you will be disconnected" approach. So anyway, no wonder the result has been a mass exodus. The truth is, I think, essentially no users understand the underlying technology and privacy guarantees of the messaging systems they're using. And in fact the illusion is enough. Everybody wants the illusion of complete privacy, even if they don't actually have much. Just tell them that they do. Tell them it's good. Tell them it uses military-grade encryption. Tell them how many billions of years it would take for the world's largest supercomputer to crack their grocery list. But whatever you do, don't mention that a simple keystroke logger would render every bit of that fancy technology completely superfluous. They don't want to hear that.

Out of curiosity, I googled the phrase "secure messaging apps," wondering what would come up, and I received a list of links: 10 Most Secure Messaging Apps - Best Encrypted Chat App. 10 Messaging Apps Comparison. The Best Encrypted Messaging Apps You Should Use Today. Best Encrypted Messaging Apps, Tom's Guide. The Most Secure Messaging Apps for Android and iOS 2020. That was hosted by AVG. And one of our sponsors actually of this show came up with the last link that I was quoting here, The Best Private and Secure Messaging Apps in 2021, brought to you by ExpressVPN.

So does any of this matter? No. As we know, there are subtle differences between apps and their underlying technologies. Through the years we've covered them all, right here on the podcast. Mostly the differences boil down to the way the various apps exchange and manage their users' keys, since the keys are everything. But here's the bottom line. Any proper use of encryption turns the conversation on the wire into maximum entropy noise, with no discernible patterns and no feasible means of decrypting what's

intercepted over the wire ever. Yes, your grocery list is safe. But - and this is the point - that is the only protection that any of these apps can provide. That's it. Period.

The only thing that any of these apps can guarantee is that while the data is in motion between endpoints, it is utterly infeasible for it to ever be decrypted. That's the only guarantee, and they all have it. But any keystroke logger defeats any of these apps, no matter how many layers of military grade encryption have been employed after the keystrokes have been logged. And of course I'm using the term "keystroke logger" because it's such a clear and well-understood term. It stands in for the concept that I want to convey, which is that on-the-wire encryption no longer means anything about our actual true privacy.

If you look at the packets moving across today's Internet, unlike the Internet of 15 years ago, when we began this podcast, all you will ever see today is packets full of noise. Just 100% random bits of noise. Today it's all noise. It's all encrypted. And no one bothers looking at it because anyone who wants to know what's in those packets also knows quite well that the encryption problem has long since been solved and that none of that noise on the wire can be decrypted. The apocryphal story of the infamous bank robber Willie Sutton applies here. The story goes that a reporter once asked Willie why it was that he robbed banks. And Willie was said to reply, "Because that's where the money is." Updating the story to 2021, a non-technical supervisor at the NSA might ask one of their top technical managers why all of the agency's work has been diverted to investments in keystroke logging, to which the technical manager would reply, "Because that's where the plaintext is."

The point I hope to make is that, barring some implementation mistake, none of this "What's the best messaging app" conversation matters in terms of its security. The real world has moved on. We've long observed that the best camera is the one you have with you. Similarly, the best messaging app is the one shared by the people with whom you wish to communicate, period. All other things being equal, I would choose the one with a clear and simple economic model that I understand and that makes sense to me. If I don't care about my metadata being monetized, about Facebook tracking everyone with whom I communicate, then WhatsApp, whose economic model clearly requires it to eventually be snooping on me, would be just fine.

But if I do wonder who's paying for this service, and if I would prefer it to be very clearly me in exchange for not having the fact of my communications being monetized, then Threema would be my choice. But in any event, the privacy provided by encryption should no longer factor into the equation because it doesn't matter at all. Anyone who wishes to have access to the content of our conversations will go after the plaintext, before it's encrypted or after it's decrypted. And I do remain uncomfortable with the autonomous key management offered by many of these systems, including iMessage. But that's also splitting hairs and really doesn't matter either. If I really cared to have a private conversation, I would never use a smartphone at all.

Leo: Yeah, I think that's fair. You said it before. Take off your clothes. Go in the middle of a field.

Steve: Yes. And then get taken away and locked up in the loony bin.

Leo: But a secure nut, and that's what's important.

Steve: Yes. Yes. Yeah, exactly. You meet with somebody you trust, leave all your electronic devices in your cars, go into the middle of a field with a big thick comforter, throw the comforter over you...

Leo: Hide your mouth because there's lip readers.

Steve: Yes, exactly. Throw the comforter over you and then whisper to each other. Any use of technology today, and your privacy is just a hope. It's just a, well, no one's probably listening. And you're probably right. Any of this is secure. Like when Rasmus and I were using Signal back and forth, well, it was cool. It's got industrial grade triple scoop encryption. But I'm just wanting to say, okay, which link was that? You know?

Leo: Yeah, there's a lot of privacy theater, I guess, in the world.

Steve: It really feels like much to do about nothing. And if you're a WhatsApp user and also a Facebook user, you've already - you're being tracked up the you know what. So who cares if your messaging app joins the community?

Leo: It's January, and it's time for the TWiT audience survey, the annual survey. Helps us understand our audience so we can make your listening experience even better. Completely anonymous, and it only takes a few minutes. So go to TWiT.tv/survey21 to take it. And thanks in advance.

That's Steve Gibson. And he's a ray of truth in a cloudy world. If you like this show, you can listen to it when we do it every Tuesday. You can actually listen live. We kind of keep a livestream going for all of our shows so you can watch it behind the scenes. We do the show Tuesday, 1:30 - shoot for, anyway, 1:30 Pacific. That's 4:30 Eastern. It's 21:30 UTC. The live audio and video streams are at TWiT.tv/live. If you're watching live, the chat room's live also, and it's a good place to go hang out with other people watching the show: irc.twit.tv.

We have on-demand versions of the show. Actually Steve's got the most interesting variants. He has a low-bandwidth 16Kb version, which doesn't sound great, but at least it's small. He also, speaking of small files, probably the smallest file is the text file of the transcript written by Steve's commission by Elaine Farris, who does a wonderful job. So you can read along as you listen or just read. It's very useful for searching, too, because you can search the transcripts and find the point of the show that you want to listen to. That's all at GRC.com, Steve's website. He also has 64Kb audio there: GRC.com.

While you're there, pick up a copy of SpinRite. If you get 6.0 now, you'll get 6.1 the minute it comes out for free. You'll also get to participate in development and the beta testing process. SpinRite, of course, the world's best hard drive maintenance and recovery utility. Lots of free stuff at Steve's site, too. It's a fun rabbit hole to fall down into and browse around.

We have audio and video at our website, TWiT.tv/sn. All the shows are at TWiT.tv, but this show is TWiT.tv/sn for Security Now!. You can also, if you listen on your schedule, asynchronously, chat with us at our TWiT forums, that's www.twit.community. We have a Mastodon instance, if you want to join the Fediverse, that's twit.social. You're welcome to join there. We'd love having you. And of course you can always get a copy of the show from YouTube, there's a YouTube

channel, or subscribe in your favorite podcast client. You'll get it automatically the minute it's available. I will see you next Tuesday, I guess, Steve. Have a great week.

Steve: Yay. Okay, buddy. Bye.

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/