SECURITY NOW!

Transcript of Episode #476

Listener Feedback #198

Description: Leo and I discuss the week's major security events and discuss questions and comments from listeners of previous episodes. We tie up loose ends, explore a wide range of topics that are too small to fill their own episode, clarify any confusion from previous installments, and present real world 'application notes' for any of the security technologies and issues we have previously discussed.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-476.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-476-lq.mp3

SHOW TEASE: It's time for Security Now!. I'm back. Steve's here. We'll talk about the big JPMorgan breach and a whole lot more security news, plus we've got 10 questions from you, our fine audience, that Steve will answer. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 476, recorded October 7th, 2014: Your questions, Steve's answers, #198.

It's time for Security Now!, the show that protects your security now, and there's never been a better time for a show like this. Steve Gibson is here, the security guru. Hi, Steve.

Steve Gibson: Hey, Leo. You know, we used to have, like, an opportunity to talk about fundamental technology things.

Leo: Yeah.

Steve: That was one of our fun sort of divergences. And now we're just panting to keep up with security breaches. And, I mean, that's really the main focus of the podcast, of course. But, boy, it really does seem like things are going crazy. And we've got more of the same this week. Just after last week's podcast, news surfaced of the largest single breach in history, which was JPMorgan Chase. They had to make a securities filing where they for the first time told the world about a problem that began in June, which they discovered in July and didn't really get resolved until August.

Leo: Wow.

Steve: And of course, while you were gone, last week's topic was Shellshock, this catastrophe that was discovered in BASH that's been around for decades. Then also we're going to talk about the return of BadUSB, the exploit code, or I should say exploit code, not the original Black Hat Conference exploit code, but some other guys duplicated that and posted it to GitHub. So it's now public. And even, it turns out, the Bugzilla bug tracker has itself a bug.

Leo: Oh, my.

Steve: So the question is, does it report on itself?

Leo: Oh.

Steve: We've got a little sci-fi miscellany. I picked up an interesting question about SpinRite that I thought I would answer in the spirit of today's Q&A. And our 198th Q&A with actually one, well, all good questions, of course, because I was able to choose among hundreds. But specifically, one really interesting one about - that you'll be interested in, and our listeners will be, I think, because one of our listeners discovered that his HTTPS switchover was blinding his advertisers to clickthroughs. And that does happen. So we'll be talking about that. And all the news of the week, as well.

Leo: Wow.

Steve: Yeah.

Leo: Wow, it's a busy day. Steve Gibson is at GRC.com. That's his website, the Gibson Research Corporation. He's also @SGgrc on the Twitter. And you have really taken to that Twitter thing.

Steve: Actually...

Leo: You didn't like it at first. I had to talk you into it. I remember this. Don't deny it. We have tapes.

Steve: I'm sure that's true because, you know, I'm...

Leo: Early days, nobody...

Steve: ...only a little bit younger than Jerry Pournelle.

Leo: This show predates Twitter, we should point out.

Steve: It does. And I've really always been a fan of non-real-time communications. I love email, and I love newsgroups. I mean, GRC runs a very active newsgroup server.

Leo: I love it, too.

Steve: And it's just it's valuable to be able to schedule that stuff when you want. But so first I was only DM'ing because I just sort of didn't understand why everybody else would want to...

Leo: Read your stuff.

Steve: Read my responses to individual people. So and then people were like, well, I can't DM you if you're not following me. And so that caused a lot of confusion. And finally I, like, said, okay, fine. And so now I'm just responding. And mostly what it is...

Leo: You're using it in a very sensible way, I think.

Steve: Well, and it's really thanks to, again, my followers, who are typically our listeners, is it's the huge dragnet. They're all out there finding things and reading things and interested in topics that relate to the podcast. And so they make sure that I'm aware of those things. And oh, my goodness, that's valuable. So I encourage it, and I read my feed. I see everything that anybody sends. It's just an absolute win for me.

Leo: Yeah, yeah. So if you want to tweet him, @SGgrc. You can also read him if you don't tweet him. Just make sure you follow him.

Steve: Yeah.

Leo: Yeah. So...

Steve: Okay. So okay. So top of the week is - and this was picked up through the news, and the SANS Institute has, like, several articles about it - JPMorgan Chase reported what turns out to be the largest breach we know of; 76 million households and 7 million small businesses had their not really critical data stolen. But any data stolen from the nation's largest bank is a problem because we want to believe that these guys really take security seriously and understand security. And so what was discovered was essentially what we're now calling an APT, an Advanced Persistent Threat.

Some attackers, and it's funny, as I'm reading these stories, I'm seeing that the standard media, and even the technical press, uses the term "hacker." And for many years I was scolded by people who were offended by my use of that term. So for a while I was saying "malicious hacker," trying to create that modification. Now I'm just much more comfortable saying "attacker." I think "attacker" is, like, the right phrase to use. So as I'm reading these I'm having to remind myself, okay, these are attackers that we're talking about, because I no longer really think of a hacker as being necessarily bad.

So under the condition of anonymity, people who are knowledgeable of the investigation, so this is even now there's - this is still under wraps, and it's all third-hand, although multiply sourced, and so the people reporting, for example, in The New York Times are sure of their facts.

Leo: So you mean JPMorgan Chase has never gone public with this?

Steve: Correct.

Leo: This is all regulatory filings that they had to make.

Steve: Yes. And it turns out that there's sort of a weird - it's sort of an odd characteristic of our current laws, which is, even this sort of an institution need not disclose if there's no clear financial impact to their customers. And they're stating that they have no evidence that a penny was fraudulently transferred from any customer accounts. So they believe that names, addresses, email addresses, and phone numbers were exfiltrated.

Leo: But not credit card numbers.

Steve: Correct. Not Social Security numbers. Not credit card numbers. And so not the most sensitive information. Oh, and not passwords. And they have seen no evidence of fraud resulting from this breach. They do believe that the attackers are based in Russia. And for reasons that aren't clear, because again we're sort of dealing with, well, don't quote me on this, but I'm familiar with the investigation that these other people say, not me personally, and like they're involved with high echelons of the Russian government.

So some people are speculating that this is as a consequence of the escalating sanctions that the U.S. government is imposing on Russia due to the whole Ukraine issue. But again, this is all speculation. Although apparently, because enough forensics has been found, they're absolutely sure that this data, these attacks came from Russian IP addresses. That much seems clear. So apparently the security team discovered this breach, as I mentioned at the top of the show, initially in July, and was able to post-date the initial intrusion to June. The attackers over this period of time attained the highest level of administrator privilege available in dozens of the bank's servers.

Leo: Not good.

Steve: No. They reached more than 90 different machines. But sort of paradoxically, no evidence of any fraud that resulted. So it's really sent a chill through the ranks of cyber-focused people in our government that this sort of thing happened. And then the next...

Leo: Was it Shellshock? Do they know how it happened?

Steve: No. And it does predate any public knowledge of Shellshock. But again, we don't know how long and whom might have been aware of the Shellshock vulnerability. What was really interesting was that they determined that a complete list of the programs and web applications being used on the servers was obtained, and that apparently from logs it looks like that this was very methodical. The Russians got a list of all the software, then went through methodically looking for unpatched vulnerabilities in all of that software. Which, I mean, so while we all agree that obscurity is not security, it's handy to have some obscurity when you can.

And so the idea that the bad guys got an inventory of all the software that was on these machines and then methodically swept it for any vulnerabilities that were known, I mean, that's a high-end attack. That's how - that's what you do when you're not in a hurry, and you're determined to get penetration. So, yikes. And the next day we heard that nine other unnamed major financial institutions have also had breaches, maybe part of this same group. It's believed part of the same group.

Leo: It's interesting that they didn't attempt to make financial transactions. Maybe they didn't have the capability. Or maybe they were just interested in kind of that basic information, which is useful and worth something on the black market anyway.

Steve: Oh, yeah. I mean, yeah, 76 million customers of JPMorgan Chase, names, addresses, email addresses, and phone numbers.

Leo: Yeah.

Steve: So yeah. You'd rather not have that...

Leo: But not Social Security numbers. Probably not enough information for identity theft.

Steve: And no passwords.

Leo: Not credit cards, no passwords.

Steve: Right, right.

Leo: Still, concern.

Steve: Yeah. And, you know, to find out that they have been in touch, that they've been into, that they've penetrated 90 servers and obtained full root level access. So the presumption is they could have done whatever they wanted to. But maybe the servers they had access to didn't have the information that would have been dumped. You can imagine that there's probably a lot of servers at JPMorgan Chase.

Leo: And, too, remember that we've known for years that this stuff happens a lot. And banks, because they have no obligation to report it and don't want to scare their customers, usually don't.

Steve: Right.

Leo: It just happened that in this case they had to.

Steve: So in breathless news we have the headlines: "Yahoo Servers Were Owned by Bash Bug Hackers." And so it's like, oh, okay, that does not sound good. So it turns out that even that headline was incorrect. Yahoo!'s CISO, Alex Stamos, clarified what happened over this past weekend. He explained that, although the attackers were looking to exploit BASH, I mean, you know, BASH, the whole Shellshock bug is still sweeping the Internet, with hackers trying to get into servers using it. He explained that they happened to take advantage of a different bug, maybe without knowing it.

He was quoted saying: "It turns out that the servers were" - and this is like a small number of servers that are Sports API servers. He said: "It turns out that the servers were in fact not affected by Shellshock. Three of our Sports API servers," he said, "had malicious code executed on them this weekend by attackers looking for vulnerable Shellshock servers." So if we read that carefully, that sounds like they were recruited into a botnet, and that is in fact the attack pattern is that we're seeing that, I mean, there are worms already. And so it sounds like something got into these servers, and it may not have been a Shellshock vulnerability, but the Yahoo! servers were recruited to look for other vulnerable Shellshock servers, which is not surprising.

He continued, saying: "These attackers had mutated their exploit. This mutation happened to exactly fit a command injection bug in a monitoring script our Sports team was using at that moment to parse and debug their web logs. The affected API servers are used to provide live game streaming data to our Sports front end and do not store user data. At this time we have found no evidence that the attackers compromised any other machines or that any user data was affected. This flaw was specific to a small number of machines and has been fixed." So this splashed headlines, I guess because Yahoo! is so well known, and the press is just frantic for Shellshock BASH bug stories. And so, you know, it's like, okay.

Leo: Fine.

Steve: Yeah, nothing there. Now, this is interesting because we've got a lot more information now. We talked, it was a couple months ago, in July, when Karsten Nohl demonstrated the Bad USB exploit, essentially. I mean, basically he put into certainly this podcast's communal knowledge the fact that some number of USB devices have rewriteable firmware, and that's a bad thing because you look at the USB device, and it looks like a thumb drive. We assume that all it is is passive storage. But in fact it's got a microcontroller in it. And now we know a lot more.

What happened is that two other security researchers last week at the DerbyCon hacker conference revealed their research. And we've talked about Adam Caudill before. So Adam Caudill and Brandon Wilson demonstrated that they had followed in Karsten's footsteps and - unlike Karsten, who has never published anything. He demonstrated his stuff just to sort of hopefully stir the industry to clean up its act. Well, these guys, Adam and Brandon, decided, you know, it's been a few months. Nothing apparently has happened. We're going to turn the heat up a little bit. So they put their entire exploit kit up on GitHub, all documented, all open source.

Leo: Oh, that's nice to know.

Steve: So what's interesting is that, for anyone who's interested in playing with this, I'm not obviously promoting this for nefarious actors. But it is interesting to know more. So, for example, we know that the USB drives which can be attacked are based on a, I guess you pronounce it Phison, P-H-I-S-O-N. Phison is one of the largest Taiwanese-based suppliers of USB thumb drives. And I'm sure they're relabeled, and in fact I know they are because, for example, the Patriot 8GB Supersonic Xpress is one that is vulnerable or exploitable; the Patriot Stellar 64GB Phison; Kingston's DataTraveler 3.0 T111 8GB drive; Silicon Power Marvel M60 64GB drive; and Toshiba's TransMemory-MX Black 16GB drive. And probably many, many more.

So their kit is, you know, they just built this, not necessarily to be widely used, but in order to sort of create some foundation. So, for example, they built it around .NET 4.0. So you have to have that installed in a Windows machine. They use Visual Studio 2012 Express, which is a free download from Microsoft. The Express versions of Visual Studio are. There's something called the Small Device C Compiler, which is an open source C compiler used for, as it sounds, small devices, meaning various microcontrollers. Among them is the 8051, which of course has been an Intel standard which I think Intel is no longer supporting, but it just sort of - it acquired, not self-awareness, but critical mass in the industry. So, like, many people...

Leo: Oh, but self-awareness would be fun, yeah.

Steve: That would be good.

Leo: Yeah.

Steve: We'll have that in the next conference. And so anyway, it's a Harvard architecture, four banks of, I think it's eight registers per bank. It's a little eight-bit microcontroller. But that's what this company, Phison, uses. And that's what these guys wrote and altered the firmware for. So they consider themselves white hats. I mean, for example, so they wanted to clarify what they did.

And so in Adam's companion blog he said, okay, what did we release? We released a patch to demonstrate the feasibility of creating a hidden partition on a USB thumb drive. So that's there. So if somebody were interested in doing that, for example, for their own purposes, they could get any of these drives and/or check whether any drives they have may be already workable. There's a utility and EXE that they provide which allows you to stick in a thumb drive, and it'll poke at it to see whether it's compatible with their firmware. And then you could play games.

They also implemented what they called a "password bypass." Some of these drives, the firmware itself supports password protection. What they did was they created a firmware modification such that it won't defeat the password if it's already there. But if you were to modify the firmware first, then it essentially neuters the subsequent application of a password that would otherwise have been supported by the firmware. So again, they're trying to demonstrate - they were sort of like trying to walk a fine line. They were demonstrating alterations to thumb drive firmware that does interesting things, but deliberately trying not to hurt anybody.

For example, they explicitly said "what we did not release." And Adam wrote: "We did not release self-replication. There's no self-replication code anywhere," he writes, "while it's possible that it could be done, and we've talked about how to do it, it won't be released here." He said: "I am confident that we, Brandon and I, could build a system that would infect PCs, then infect a significant percentage of thumb drives, and then infect other PCs. But," he says, "But, and this is a big but, what we released doesn't make that easier in any significant way. Your average script kiddie will never be able to do it. There are only a small number of people" - I think he underestimates that. But he wrote: "There's only a small number of people that would be able to do the work needed to pull that off. Those people could already do it before we released what we did."

He said: "The threat of this happening is the same as it's always been." So they're obviously sensitive to claims that they are escalating. And I would argue a little bit. I would say, you know, the more of this kind of work that's out there, it facilitates more than if it weren't there. And certainly they're...

Leo: They're just kind of showing off. There's no real value, is there, to publishing this? Do we learn anything from it?

Steve: Yeah. I think maybe it keeps it in the air. And he ends by saying that what we're hoping for is that manufacturers will add code signing [audio dropout] to prevent future modifications.

Leo: So it's kind of like Firesheep. It's like, well, let's set the whole thing on fire, and then they'll have to do something.

Steve: Yeah. I mean, and he also acknowledges, I don't remember now if it's on GitHub or in his blog posting, but somewhere he says it's clear that all of these devices are already out in the world. So if a change were made tomorrow, and who knows if it's even ever going to be made, then if it were made, it would still take a decade for, like, all of the existing drives to fill up and die or go away or become obsolete. So essentially we have a problem.

And anyway, so again, I'm responding to sort of overwrought press stories saying, oh, my god, BadUSB is back and now it's public. It's like, yeah, okay. And essentially what Adam has written is saying we were curious, so we poked at it, and we were able to do it, too. And I guess his point is it is reproducible. It's not difficult. And by us doing this, we really haven't changed anything because, I mean, and he is right. You'd need to be able to write 8051 assembly language, or C, I guess, but disassemble the firmware, reverse engineer it, figure it out, make modifications, and then move forward. So it's a project, but certainly not beyond state-level actors. That would be trivial for them.

Now, again, bringing a little bit of sanity to the Bugzilla report. What the press said was that a "Bugzilla Zero Day Exposes Zero Day Bugs." So here's the story. What was found was that a non-default configuration of Bugzilla could allow privileges based on email domain. The default installation doesn't do that. But it's certainly possible for some Bugzilla administrators to say, hey, we want - we're just going to make this easy for ourselves. Everybody with a Mozilla - say, take the case of Mozilla - Mozilla.org domain email address just automatically has privileges to see bug reports for submitted zero-day, known-to-us vulnerabilities that are obviously not available to the public.

So the point is that Bugzilla is going to have administrative privileges that allow some people to see submissions that are non-public. It turns out that it - so Bugzilla can be configured to give those privileges by email domain. And the problem is, when you create a Bugzilla account, it does no email verification. So anyone could create in this example...

Leo: Oh, that's nice.

Steve: ...yeah, a Mozilla.org email address and instantly have...

Leo: It's a bug.

Steve: ...admin privileges.

Leo: Okay, that is a bug.

Steve: It's not a good thing.

Leo: That's stupid. That's easy fixed, easy to fix, though.

Steve: Always been there. Apparently, like from the beginning, it's just it never did email loop verification. And maybe the people who understood that weren't worried because it isn't the default condition to give blanket permission by email domain, but it is an option. And some people apparently have used that. And if those companies were using Bugzilla in that way, the fact that there's no email account verification means obviously that, if sensitive bugs were being submitted, people you didn't intend to have see them could access them. So that's what that story is about.

I wanted to note, I tweeted, I think it was yesterday Amazon sent me a note that they had shipped my copy of "Edge of Tomorrow," which was fabulous. Just Tom Cruise has been doing a bunch of great sci-fi movies. I loved "Oblivion" from summer before last.

Leo: Wait a minute. Shipped it? You didn't just watch it on demand or stream it? You got a physical disk?

Steve: Oh, no. I like - remember? Okay. Leo, I'm just barely using Twitter.

Leo: Were you waiting by the mailbox with bated breath? My disk is here, my disk is here.

Steve: I've seen the movie. I loved the movie. And so I'm looking forward...

Leo: Oh, you wanted to own it, yeah.

Steve: Yeah, exactly. I do, I like to own the bits of, like, the physical - like I said, I'm still a little bit old school.

Leo: You don't plan moving anytime soon, so...

Steve: No, no. Exactly. So I'm looking forward to seeing it again. I just wanted to give our listeners a heads-up. Many people responded to my tweets in agreement. IMDB gave it an 8 out of 10. Rotten Tomatoes gave it a 90% with an audience score of 91%. It's just a great - unless you really have a problem with Tom Cruise is the only reason I could imagine that you wouldn't just love it. So I wanted to let our sci-fi listeners...

Leo: I'm not crazy about him, but it sounds like he's pretty good in this.

Steve: He really is. Actually, he did, as an actor, he did a great job. At the beginning of the movie he's like this PR flack, and he realizes, okay, I guess I'd better start shooting things. Anyway, it was sort of time travel and wonderful stuff. And, oh, and are you a "Homeland" watcher, Leo?

Leo: Love "Homeland." I didn't, now, don't - no spoilers because I haven't watched the season premiere yet.

Steve: Oh, my lord, Season 4 began on Sunday. And, oh, it looks like it's going to be - I've been exchanging texts with Jenny because she's equally into "Homeland." And I think the series has really hit its stride. You have to allocate an hour and 45 minutes because it's a two-episode premiere for Season 4. But, wow, it just really looks good.

Leo: Good.

Steve: And in keeping with the Q&A spirit of the podcast, I ran across a question from Martin in Frankfurt, Germany, wondering how much use he might be able to get out of a troubled drive after using SpinRite. And I thought I would use his question and answer that for everyone. He said: "Hello, Steve. I have a question about how safe it is to still use a drive after it was SpinRited." And he said, "Yay, official new word since the last Security Now! podcast." I guess someone said SpinRited or SpinRitten. Anyway, you can make up your own word. "And SpinRite has found sectors that are not recoverable."

And that's what makes this question interesting. He said: "In such an instance, does SpinRite tell the drive to map out those bad sectors so that they will never be used again? Is it therefore safe to still use the drive and the remaining good portion? Or should one consider not using the drive at all anymore since it shows rather severe damage? I guess it all depends on the overall age and usage of the drive and how important the data you want to put on the drive is to you. But just from a purely technical point of view, would SpinRite move those bad apples out of the way for you? As always, thanks for your great wisdom and advice."

Okay. So here's - that's interesting because what happens is a drive will never relocate a sector that it is unable to at least finally read successfully. So when SpinRite is doing recovery, it is trying all sorts of tricks, as I've talked about in the past, to get just one last successful read of the sector because oftentimes what's happened is that the error is just a little too long for the error correction to be applicable. So if we can just shorten that by a bit or two, then the ECC can correct the sector, the drive apparently internally breathes a huge sigh of relief and is then willing to relocate the sector. That is, it says, wow, you know, here's your data. I'm putting a good sector back in here.

Leo: Whew.

Steve: But it will never do that unless it's able to read it correctly. So what SpinRite will finally do, if it exhausts itself and is completely convinced that no force on Earth will ever be able to read this entire sector again, is SpinRite will settle for the uncorrectable data on the theory that oftentimes that's still way better than getting none. You might lose, for example, 12 bits where the drive could have corrected 11, so like two bytes, or maybe three depending upon how they straddle the byte boundaries. But you get the other 512. And for example, if that's a direct resector, suddenly you go from nothing downstream of that point in the file system being accessible, to probably everything else being accessible. So massive win in some instances.

But then SpinRite will rewrite the sector. And then reading it subsequently allows the drive to then relocate it and take it out of service. So again, there's, like, a lot more going on that most people appreciate, which is part of the reason that SpinRite achieves the magic that it often does. There's a lot going on behind the scenes. So that's the story. I would say, if you've got big red uncorrectable U's, which is what shows up on the SpinRite map, that's a drive which is really doing everything it can to tell you, yeah, you know, don't use me anymore.

Leo: Mm-hmm.

Steve: What you would have to do really is, if you had to use it safely, would be to use a file backup, after running SpinRite on it to get off all the data that you can. Use a file backup to move the files - again, this is if you have to use the drive for some reason. Like you're on a desert island and...

Leo: That's it. There's only one drive.

Steve: There's no Fry's and no Amazon delivery. That's it. You have to use this drive. You're a remote descendent of Robinson Crusoe. So move all the files somewhere else, I don't know where you would put them if there's no other drive, but put them somewhere. Then you would have to do a full file system format so that the file system, that is the OS, could itself discover those bad sectors and mark those out of the file system so nothing ever tried to use them again. Then put everything back.

But again, one thing to ask yourself is why are those sectors absolutely unreadable? Did the drive get dropped? Did it get bumped while they were being written? Because that's a possibility, where the drive's okay, but the head was made to wobble when it was writing them, and so that's one of the things that of course SpinRite does is it rewrites the sector and then checks to see whether it was just a miswrite or if there's actually a defect there, in which case the drive can see it and swap it out to safety. So again, little more going on behind the scenes than is always obvious.

Leo: Very interesting. We've got questions. You've got answers. Our listener-driven potpourri #198 starts with Simon in Melbourne, Australia. We're going to stay in the commonwealth for the moment. He reminds us that HTTPS everywhere has a kind of a negative side effect: Just a quick note for the show. A while ago you suggested TWiT use SSL. While I agree with you that HTTPS everywhere is a good idea, it does leave caching proxy servers such as those used in universities and some ISPs useless, resulting in more bandwidth for TWiT.tv and others. That's actually, well, okay. I'll keep going, and then I can talk about this.

Steve: Yup, yup.

Leo: In a non-SSL download, one student in a classroom would download a podcast, then 29 other students would download again, but these subsequent downloads would be sourced from the caching proxy either at the ISP or university gateway level; whereas, if SSL were implemented at TWiT.tv, all 30 students would be hitting the server individually, and no caching would be available. Oh, he's right about that. I'm sure you're aware of this, just wanted to bring it up. Great show, by the way. That's right; isn't it?

Steve: Yeah. Well, except that you're using a CDN, and these days I'm surprised by, when I look at links from even small sites have independent content delivery networks that are, like, sourcing even relatively small documents and things. So, and typically CDNs will also be supporting HTTPS, or have that available as an option. But he does make a point that I think is a good one.

For example, I remember well when I was implementing ShieldsUP! that I needed, I explicitly was establishing an HTTPS connection to avoid ISP caching proxies, in order to get a connection to the user, in order to obtain their true IP address. Because, for example, Cox Cable here in Southern California, they run a caching proxy. ISPs do that because, well, for two reasons. If there is content out on the Internet which is being pulled by their customers multiple times, like say all of the little images and twitches and buttons and stuff on Amazon's site, that stuff appears on multiple pages all over their site.

So the idea is that the first of the ISP's customers who visits that page, their queries go through that caching proxy. It essentially holds their request. It issues the request on their behalf, obtains that resource and caches it, and then returns the results. The beauty of that is then all of the other customers of that ISP, when they make the request, the proxy is able to intercept it and reply locally. It's the locality of the cache that ends up meaning that it's a huge speed increase for the ISP's customers because they're having, you know, basically it's like a huge portion of the Amazon remote network has moved right over onto their LAN, right onto their ISP network, by caching those resources.

And then the other reason the ISP uses it is that, on some level, the ISP is paying somebody for its own bandwidth, that is, the ISP's use of Internet bandwidth. And so if they cache within their network, then, for example, in Simon's example, one blob of podcast gets transferred into their cache, and then 29 other transfers are avoided by serving it from the cache. So not only does it lower the load on the ISP, it lowers the ISP's network transit, also. Which on some level they've got contracts for and they're being billed for and everything, as we've been talking about.

Now, HTTPS, as Simon says, short-circuits all of that because the whole concept presupposes a non-authenticated connection. That is, the user's browser thinks it's actually connecting, in our example, to Amazon. It's not. That's why it's called a "transparent proxy." There's no actual proxy protocol negotiation. It's transparent. The connection appears to be made to Amazon. It's actually being intercepted by that caching proxy which is then making a separate connection to Amazon. And as we know, that can only happen, you can only get a transparent interception, I mean, really it's a man in the middle. It's not a man-in-the-middle attack, it's a man-in-the-middle proxy cache. But it's still in the middle. And that's what HTTPS prevents because it's going to verify the certificate of Amazon when it terminates that connection, and the ISP's proxy is unable to do that.

So anyway, it's worth - we're going to talk - there's a couple more questions that have been raised about this as a consequence of the topics we've been covering on the podcast recently, some interesting consequences of, like, switching the world over to all HTTPS. And one of them is that the benefit that was available to ISPs and their customers does get lost, all of the assets that have to come from that source.

Leo: I can live with that, though. I mean, that's, you know.

Steve: Doesn't hurt me.

Leo: We don't mind. Your pages are small.

Steve: I haven't seen - yup.

Leo: Yeah. Jamie Eastland, Marlborough, Massachusetts notes that users may be not always able to count on pleading the Fifth. This must be something from last week. Well, yeah: In your discussion with Father Robert last week, you noted that people cannot be compelled to divulge passwords, unlike physical keys. I only wish that was true.

The Massachusetts Supreme Court recently ruled that defendants can be compelled to provide passwords under certain conditions, such as when doing so would provide the prosecution with little or no information. Okay. In this case the defendant had already admitted ownership of the computers. He had knowledge of the password. Knowledge of the password can be used to assert ownership of the computer, but in this case the ownership was already admitted. So the court assigned no value to the information encrypted on the hard drive and rationalized that the defendant could be compelled to produce the key.

Seems like a shaky logical construct, he writes, but that's the rule in the commonwealth at this time. That might be overturned at the federal level, I have to say. Keep up the wonderful work. I've been a listener and a SpinRite owner since year two of the podcast. My commute is an hour each way, so while I wish you produced more than 90 minutes each week - don't egg him on, please - I'm just as happy that there isn't enough news to warrant more frequent podcasts. That's an interesting story, actually.

Steve: It is weird. And I dug in a little bit, just like saying, what is going on? And I'm more puzzled having done that than I was just sharing what we have with our listeners. So...

Leo: I think what you were referring to was the well-known - this came up when Touch ID first came out, the well-known legal fact that in fact a physical thing like forcing you to use your fingerprint to unlock your phone is legal.

Steve: Yes.

Leo: If you have a warrant. But you can't force somebody to divulge the contents of their brain.

Steve: Right. The idea is it is under the Fifth Amendment it's considered self-incriminating to require someone to basically testify, it's considered testimonial information against your own interests, which we're protected from with the Fifth Amendment. And we've talked on this podcast about this several times. This is sort of it's common belief that that is the case. And it was definitely the case that a court in Massachusetts said, eh, we don't think so. I mean, the Supreme Court in Massachusetts said that, which is a little worrying. And as you say, Leo, I wonder, if this were taken to the U.S. Supreme Court, what would happen. Like if that would stand.

But I read more about it, and it really did seem like the logic was bizarre. It was we already believe that we know everything that we're going to find out from you decrypting the contents of your hard drive. So therefore we can compel you to do so. Well, it's like, wait a minute. If you're convinced you already know it, and I don't want to share it, then how does that make any sense?

Leo: Presumably if they uncover something they didn't know, it's inadmissible. I don't know.

Steve: Yeah. It was strange. Thought it was just worth mentioning because we were assuming that we had protection, and maybe not.

Leo: Right. Chris in Sacratomato [sic], California is trying to help law enforcement: While I was patching several servers at my company for the Shellshock bug, I decided to set up a few loggers to see what sort of commands people were trying to send to our servers. I found several people were trying to use curl or wget to download malicious code. In some cases - and what he's saying, we should explain, is that they would log into the server, then run curl or wget to put malicious code on the server. In some cases I downloaded the code to see what it did, and I found scripts designed to connect a computer to a botnet and perform TCP and UDP flooding when given the appropriate command through IRC.

My question is, is there somewhere I can send these programs to help stop these people? The programs contain IP addresses of the IRC servers for the botnet, so the right person could potentially just go shut down these servers. I looked online and found an FBI cybercrime page. But when I called the field office they said, eh, we don't care. Eh, you can have it. We don't care.

Steve: I love this question because, first of all, I love Chris's team spirit.

Leo: Good for him, yeah.

Steve: But, yeah. But the fact is, and I'm very sympathetic to this, law enforcement is so overwhelmed, especially at this moment with Shellshock, I mean, the fact is all server logs are all full of this. And so it's like saying, you know, in a middle of a monsoon, hey, look, I have a glass of water. It's like, look, you know, just put your hand out. Yeah. So the problem is the only way the FBI would get involved would be not preemptively like this, but it would be if a company was able to demonstrate damages. And I once knew, there was a dollar amount, I think like $10,000 of damages. So there's an amount of money you have to be able to demonstrate you have been cost by the malicious actions of somebody else. And then they create a case, and then someone's assigned to it. I mean, so it's a really long and involved process.

So unfortunately, while it would be nice if they were in the business of accepting logs from people, at this point they're seeing potentially malicious activity, but not evidence of a crime. And so unfortunately the bar has been raised to the point now where - oh, and the other thing I remember is that they're already typically busy solving crimes of millions and tens of millions of dollars. And so you're also ranked on the economic loss scale. And even if you barely clear the threshold, you're still way down in the pecking order. So unfortunately the reality of where we are today is far less ideal than we would like it to be. But given the resource constraints, it's the way things have worked out.

Leo: Hey, call them up and say somebody stole my iPhone, see what they do. Nothing.

Steve: Yeah.

Leo: You can file a report, sir, and we'll be glad to file the file. Gary N. in Kalamazoo, Michigan brings up an important point: All right, I realize I'm being a little picky here. But just to set the record straight, you said that Ethernet should probably not be used to connect a 400-foot run. But technically you are still using Ethernet, it's just running over a fiber optic cable. It always drives me crazy when people call twisted-pair wiring "Ethernet cables." Ethernet's a protocol. It can run over coax cables, twisted-pair copper, fiber-optic cabling. It can run over pig spit if we can figure out how to do it electrically. It's all Ethernet.

Steve: Yes.

Leo: I added the pig spit.

Steve: Yes.

Leo: That's editorial on my part.

Steve: But it was, it was good.

Leo: It's a good point. It's a protocol.

Steve: So annoyed with myself for, like, being lazy that way because details are important, and Gary is absolutely right. As you say, Leo, Ethernet is a protocol. It started back in the, what, it wasn't 10, is it 10Base...

Leo: 10BaseT.

Steve: ...2, I think we would call it, 10BaseT.

Leo: Or 10Base2 because "T" was twisted pair.

Steve: That's right, yeah. So it was 10Base2. And then, or, no, no, it was - wait. 10Base2, then 100Base2, and then...

Leo: I think we used also thin - we used coax. I didn't use thin fiber [indiscernible] coax, yeah.

Steve: Oh, yeah, that was all coax. Anyway, the point is that it is inherently - and this is what Bob Metcalfe invented at Xerox PARC, where you would use - it was a shared medium technology. Everybody would clamp on to the same coax. And it was called, what, CS something CD. I haven't used the acronym for so long. Like, oh, Carrier Sense

Multiple Access Collision Detection (CSMA/CD), that's what it was, the idea being that everybody would listen. And if somebody wanted to speak to somebody else, they would wait for a pause, like when the shared cable wasn't carrying communication, and then just launch in. Just send a packet out onto this cable using the Ethernet protocol. And the idea was that, if there was a possibility that two people might both start talking at the same time, as often happens on the podcasts...

Leo: We need a collision-based podcast.

Steve: And so you'd have collision detection which happens with, exactly with the podcast, both people talk and go, ooh. And then they both stop, and then they wait a random amount of time and start talking again.

Leo: They time out for random interval, yeah.

Steve: Exactly, and then start talking again. And so, and that's exactly how the Ethernet protocol manages to operate on a shared medium. And as you said, Leo, we started off with literally tapping a coax. Then, for example, we have the twisted pair connection where we have, instead of everybody tapping on the same cable, because the twisted pair - you can't just tap into a twisted pair. So that's a point-to-point topology electrically. But the hub, when you have a hub and not a switch, the hub retransmits out to everybody. What anybody sends in comes back out. So it's essentially a point-to-point electrical connection, but it's a shared information connection.

So again, Ethernet works in the same way. When you have a switch, there's intelligence there which is able to learn which MAC address is connected to which port of the switch. And you can have multiple MAC addresses that the switch learns, or on different ports. So there, that's a store-and-forward technology where a packet comes into the switch, and the switch has the ability itself to monitor the protocol on the wire and learn, by memorizing the ARP transactions happening, who's down which wire, and so it's able to forward the packet only onto the connection that's necessary.

Now, that has the advantage in a large network of allowing substantially more traffic to transit if it's dispersed traffic because the switch sort of functions like a traffic cop, sending the data only down the wires where it needs to go, rather than essentially flooding the entire network with every single sender's data across the whole network. So again, thank you, Gary. You're absolutely right. I stand chastised. I'm annoyed that I - because that's the kind of distinction I really think this podcast should be making.

Leo: But Steve, how does Token Ring work?

Steve: Completely differently.

Leo: No, no, no. No, no, aghhhhhhhh. Moving on. Alex in Orange County - actually, this is a long one. Get ready. Brace yourself. Because he has noticed a potentially negative side effect of switching to SSL. Another one.

Steve: And Leo, you're going to care about this one.

Leo: I do already, deeply.

Steve: Yup.

Leo: Hi, Steve and Leo. Been listening for a little over a year now, filling my commute time with the show. Thank you for that. I know you have brought up how there is more and more of a push for sites to switch over to TLS and SSL. Really the push is coming from Google. I actually switched my site, a hobby site with about 5,000 visitors a day - that's nice.

Steve: That's a big hobby site.

Leo: That's a big hobby - to use TLS with, might as well, Perfect Forward Secrecy, right on, in the first quarter of this year to help keep my visitors' forum passwords secure. They may use the same passwords elsewhere, after all. In the last couple of months I had a couple of my direct advertisers contact me about how clicks to their site from mine have dropped drastically recently. One of them was about to not renew their banner ad. What I realized after the first one was they weren't getting the referrer data. That meant that my site wasn't receiving credit for sending my visitors to them. The visitors were going, but the packet didn't contain referrer data.

At first, I assumed it had to do with how I redirect users instead of the link going direct to the target site, and the advertiser accepted that and the stats I provided. But after a second advertiser contacted me about the same issue, Google Analytics was not showing much traffic from my site, I decided to look into changing how my direct banner ads work with a little JavaScript to keep my ability to track stats. I started with a simple page, one basic "a" tag with an HREF to another of my sites. When I tested this baseline, the referrer header was missing. WHAT??? That's how he wrote it, actually. WHAT???

So I did some searching to discover that the trouble was that web browsers do not pass the referrer from sites that are using SSL. Oops. It makes total sense, if you think about it. Browsers assume that the URL of the source may have some sensitive information that shouldn't be shared with the target site - not, of course, the case with my site. What I also found is there is a meta tag that will tell the browsers to "pass the referrer for links on this page." I have added that to all my pages, and suddenly even the redirect method I was using for my banner ads was now passing referrers of the original click source. Bravo.

Anyway, I realized that many others probably don't know about this side effect of switching to SSL. I've worked in security for a decade, and I wasn't aware of it, not were a couple of peers in the industry I asked about it. If the topic of having people switch their sites over to SSL comes up again, it might help out a few of your listeners who choose to do so and might have a negative impact from this little side effect. Thanks for reading. Thanks for providing the show. Been a big fan of Leo since the early Screen Savers days. Your neighbor in Orange County, California, Alex.

Steve: So, okay. Things are a little more complicated than Alex indicates, but he's essentially correct. It's been part of HTTP from at least the 1.1 spec, in the RFC. And the RFC states that a web client like our browser should not include the referrer field in a query made from a secure domain or a secure page, when it's on a secure page, to an unsecure domain.

So it's the security crossing that is the concern. And the argument, as Alex suggests, is that it's certainly possible that somebody is using HTTPS, and because they're using HTTPS on their own site, they feel comfortable passing security-sensitive information in their URLs. I mean, that's still a bad idea because you just don't want to do that. All kinds of things cache that. And, I mean, even your browser will keep a list of pages you've visited recently. And while it may only be for you, that's still sitting in your computer. And putting things like usernames and passwords in the URL has been deprecated now for quite a while from a security standpoint. So it's a bad idea.

But back at HTTP 1.1 they were saying, look, maybe somebody is doing that. Which you could argue they could do securely over HTTPS because that's not going to be sent in the clear. So if they were on a page that had that sensitive information in its URL, if that was part of the page's address that, like, that brought the user to that page of a site, and that user clicked on, for example, an ad link that was going to some other domain that was not secure, then strip out the referrer header.

Well, the problem, of course, the way the industry has evolved is it's the referrer headers that tell advertising sites where the ad was clicked on. That's the way the advertising servers are able to credit people's sites who carry ads with clickthroughs to the ads and generate revenue. So if the links we're clicking on are HTTP links - so the first way to solve the problem is for the ads themselves to be served over HTTPS. Then, even though they're a different domain that's secure, at least they're both secure. In which case the browsers will leave the referrer information there.

But the more robust solution was an addition made in HTML 5. And it's a tag that, if you know that your URLs are clean, that is, that no URL on your site is going to be leaking sensitive information, you don't care if that URL is known to anybody else, then you can add a meta tag and, like, stick it in your include file for the headers on your site, and bang, now suddenly all of your sites have it. The meta tag is meta and then name=referrer, with two R's and then ER, so I guess that's three R's total.

Leo: Are you sure it's three R's?

Steve: There's a famous typo.

Leo: Isn't it misspelled?

Steve: It's misspelled in the header, but not in the meta tag.

Leo: Not in the meta tag, okay.

Steve: Yes, because they made a famous misspelling in the spec.

Leo: One R, R-E-F-E-R-E-R, yeah.

Steve: Yeah. Okay. So and then the content is one of four values. So you can say "meta name=referrer" in quotes, and then content equals either never, which is a way to tell the browser that your URLs are sensitive, and you never want them sent out in the user's queries originated from one of your pages. Or you can say - you can use the keyword "origin," which for example might be you run a search engine and want people to know that the traffic came from you, but not the query tag. So there the browser will strip off after the - at the end of the URL before the question mark and the so-called "query tail," that part gets removed. So only the page ID, but nothing afterwards will be added. Or you can use the tag "default," in which case you get this behavior, where it tries to be smart, where it'll add the URL if it's HTTPS that you're going to, but not if it's HTTP. Or you can say "always."

And so if you know that you don't have sensitive information in your URLs, and you do want to make sure that clickthroughs to nonsecure sites receive the referrer information, so that you get credit, then you would want to say "always." And that modifies the behavior of compliant browsers. Today, Google Chrome and Safari both support this meta referrer addition up in the header. Firefox doesn't yet, but it's in progress. And IE doesn't at all. So very useful information. Thank you, Alex, for reminding us of that.

Leo: All right. Continuing on, Steve, more questions from our fabulous crew. By the way, you can leave questions at Steve's website, GRC.com/feedback. This comes to us from Chris Avelleyra in Fort Dodge, Indiana. He's worried that MAC address randomization might be a problem. Now, you talked, I bet, last week about Apple's bogus MAC address randomization.

Steve: Yes, exactly. Yup.

Leo: He says, in fact, he's glad you talked about it, or the lack thereof as the case may be, last week: I have a question. I was actually dreading the feature for the following reasons: I use MAC filtering on my wireless network, so I was concerned that my phone would never automatically connect when I was home and I would have to turn that feature off on my router - I'm trying to learn as much as I can from your podcasts, thank you for the awesome free service - so I'm not clear if MAC filtering is actually real security. Hey, but I figured every little obstacle would at least prevent the babysitter from logging in, even if she miraculously found my hidden network and guessed the impossible passcode.

Is MAC filtering really a viable, valuable security feature? If so, well, what happens if you get your wish and iOS truly comes up with a different MAC address every time I try to connect to my home network, or any other network with MAC filtering where my device is allowed, such as a BYOD-enabled work place? As crazy as it sounds, I might actually want the ability to turn off the randomizer feature. I'm probably one of the few that really doesn't care if I'm being tracked. I'm pretty boring, got nothing to hide. Your insights appreciated, as always. Chris.

Steve: Okay. So in the first place, there's nothing wrong with MAC address filtering. But it's a little bit like locking the door and hanging the key next to it. Because anyone who is going to hack you, who wants to get onto your network, will be looking at a packet sniffer. And they may realize pretty quickly, when they're getting no response at all from your router, that there's probably a MAC address filter. All they have to then do is see any other device on your network which is getting a response from the router, and use that MAC address.

Leo: Spoof it. Spoof it.

Steve: The problem is the MAC address is not part of the encryption. It's part of the transport. And transport is necessarily outside of encryption because that's the protocol. So the problem is, anybody who's looking will see MAC addresses that are valid and passing through the filter, and they can use it. So, yes. First, so the first part of the question is, eh, you know, it's not bad to have it if you're someone who likes twiddling with technology and with router configuration. And of course this means that every time you add another device, you've got to go through permitting it.

Leo: Now, that's the problem. It's a pain.

Steve: Yes, it ends up being a lot more to maintain.

Leo: He's also doing, it sounds like, SSID hiding, which is equally worthless.

Steve: Yeah. Yeah. So the second part is that the randomization, if and when they actually do make it useful, should not be a problem because the phone knows, if it's passively sort of just like probing and looking at WiFi ports, you know, WiFi hotspots, or if it is at your home, and in the past it has connected to yours, and it knows you and yours, and in that case when it's actually making a connection it always uses its proper real fixed MAC address. So I'm very sure that filtering won't cause the iPhone to have problems.

But also, for what it's worth, Chris, eh, it's not clear that filtering is really buying you as much security as it is sort of annoyance, exactly as Leo says. Every time you want to bring up a new device on your router, you've got to go figure out its MAC address and add that to the table. And anybody who is looking at what's going on in the air can see all the MAC addresses that are being permitted.

Leo: And SSIDs, by the way.

Steve: Yeah.

Leo: Patrick, Laramie, Wyoming. He has a question about Ethernet MAC addresses in general: Steve, why doesn't every device use a completely random MAC address? Why even bother to have a fixed, unique MAC address? It doesn't make any sense. Each MAC address is 48 bits, which means if you generate a MAC address randomly, you have a large but non-infinite chance that you'll generate two identical MAC addresses. It's one in a large number. You think I'm going to read that?

Steve: 2^48. 2^48.

Leo: Oh, yeah, that's not so bad. It's 281,474,976,710,656. If you care.

Steve: Yes.

Leo: However, MAC address collision is detectable. That means that in the case that two devices join the same network with the same MAC address, a collision would be detected, and one or both of them would just generate another completely random MAC address. The statistical probability that both devices continue to generate the same random MAC address rapidly and aggressively approaches zero. Problem solved.

Why aren't we doing this? Is there any reason whatsoever that a device needs a globally unique identifier at the MAC address level? I cannot see one at all.

Steve: You know, it's just historical.

Leo: No, but also it's - you don't want to have that kind of computation necessary on a dumb device, a cheap, dumb device. Why do that?

Steve: Well, and that's just it. When Ethernet, the protocol Ethernet, was created, remember those network adapters cost thousands of dollars. It was several thousand dollars to put one computer on the Ethernet. And this was - it was discrete components. And as we've talked about this before, the way the MAC address is segmented is that the first 24 bits, the first half, the left half, is a number given to the manufacturer of Ethernet protocol adapters. And then the right half they control. That's their own serial number of adapters within their manufacturing number.

So it was deliberately - and that was sort of a clever approach that Metcalfe came up with to specifically be able to give everything a fixed address and have zero probability of collision. Today, in this day and age, we toss around the idea of, oh, detect a collision and make up a new address because we could do that trivially with the technology we have now. But back when this standard was created, that would have cost money. I mean, it was just like enough that it wasn't even - didn't even begin to be worthwhile.

I actually agree that, in this day and age, it really no longer makes sense, that having a random MAC address would be workable and feasible and cost nothing. It's not totally true that it's easily - that a collision is easily detected. You will detect a collision, but it's not in the - it doesn't have the robustness of, for example, packet collision on a shared medium Ethernet wire, where the system is designed for that. The Ethernet assumes non-collision MAC addresses. And, I mean, it makes it very unhappy when there is a collision. All kinds of thing sort of stutter and lock up and stop communicating. And while that could probably be changed, it's like, well, we have a standard, and nothing's going to happen to change that. But for what it's worth, as sort of a thought problem, I think it's entirely feasible, if things had sort of come out differently.

Leo: There's some value to having static MAC addresses, I think.

Steve: Yeah. And it's not going to not...

Leo: I mean, people are worried about the privacy issue. But I think there's some value to it. Howard Matthews in Birmingham, which is still in the still United Kingdom. He was worried about Scotland, I think. He's wondering whether higher HDD capacities intrinsically mean faster throughput. Listening to your discussion about sector interleave got me thinking. Always dangerous. I know access times on hard drives are limited by the fact that the heads have to move to the right track. But once there, the higher the data density, the faster the throughput; right? So these new 8TB drives ought to be faster in use, let's say, than a 1 or 2TB drive because they've got higher data density; right? Right? Right?

Steve: That is actually correct. There are three ways you can increase the density. You can create the linear bit density, which is the number of bits around the circumference of individual tracks. You can increase the track density, which is the inter-track spacing. You lower that, and you get more tracks per inch. And of course you can add the number of platters. A platter is going to have two surfaces, so if you have more platters, you have more surfaces. Only one of those three things, the actual linear bit density, directly relates to data throughput.

But I remember looking carefully at this when I was working on 6.1, and I'll be of course coming back to it as soon as SQRL is behind me, because I was curious. Are we, like, where are we in approaching the six - now I'm trying to remember what the nomenclature was. I know there's a three and a six in the SATA spec. And I guess it's maybe - is it 6MB?

Leo: Gigabits. It's gigabits.

Steve: Yeah, okay. So drives are still way slower than that. So we have headroom before we start saturating the interface between the drive and the motherboard when we're running at absolutely full speed. Which I was glad for because we seen to keep increasing the density. And of course now that we've switched to helium-filled drives, that reduces the flying height. That allows the heads to come closer. And if the heads are closer, they essentially have more resolution.

And so that'll be the next thing that increases density is lower, closer flying heads because there's a helium bearing rather than an air bearing. And that also turns out that air was creating a lot of friction. Helium is lower friction, and that allows the discs to spin faster, as well. So we just keep pushing these things in ways that are sort of amazing. But it is the case, yes, higher density means, like as a general principle, means higher throughput because certainly one of the ways they're getting more density is packing more bits per inch around the circumference.

Leo: We used to say, remember the days of Stacker?

Steve: Yeah.

Leo: We used to say, oh, well, Stacker offsets the extra overhead of compressing the data by making throughput faster because the data - I don't think that makes any sense, but that's what we said.

Steve: We were looking for ways to speed these things up.

Leo: We just were trying to justify it. Steve confused Mary the IT Girl in Zephyr Cove, Lake Tahoe about the strength of public keys. Shame on you, Steve.

Steve: Oh, well.

Leo: You recently said that Elliptic Curve Cryptography is both stronger and faster than RSA, with far fewer bits in the key. I'm having difficulty with that concept. If brute force is the only known attack, how can fewer bits, combined with a faster algorithm, provide more strength? Hmmm?

Steve: Huh. So I understand. I mean, that makes sense, Mary, that that's confusing because we're talking about different technologies. In symmetric encryption, where there is a symmetric key, which is unknown, the way you crack that, given a cipher which doesn't admit any clues, is brute force. And so it's 2^n, where "n" is the number of bits, and there's that many possible keys. And assuming a random key and a brute force at random, it's going to take on average half of 2^n, or 2^n-1, guesses in order to - because all you can do is guess at the key.

That is not the way we crack asymmetric, that is to say, public key encryption. With public key encryption, for example, we have a very different problem. We've got, in the case of, for example, RSA, we have two prime numbers. And multiplying them is easy. Factoring the result when we don't know what the two prime sources were is hard. And so that's the secret in classic RSA crypto is that we have a problem where - typically in crypto it's called a "trapdoor." It's a one-way function. It's trivial to multiply those.

But then when you look at the product, it turns out - so, for example, you don't brute-force that. You perform a prime factorization of the product of primes, hunting through a vast possible space for the two numbers, both prime, that were multiplied to create that product. And that's a really hard problem. In elliptic curve crypto, we essentially have a discrete logarithm problem on a specific curve. That is, there's a mathematical curve that is described, and then it exists in what's called a "finite field," meaning that essentially we do a modulus, and we only keep the remainder of the values that we're computing within this field. And when you don't have the whole value, you only have sort of a hint of it, that's a very hard problem.

But again, it's not something you brute force. There are various ways that cryptographers have of attacking this from characteristics of the encryption, but it's not just sitting there and saying, oh, I've got this many bits, so let's start guessing. So the confusion arises from the fact that, in one case, brute-forcing is done with symmetric ciphers, and the way you crack asymmetric ciphers is completely a function of the nature of that cipher.

And so what I was talking about with ECC is that, relative to RSA, is that because ECC is harder, that is, we can take problems with very few bits and, for example, like multiply 17 and 13, and then easily figure out what the prime factorization of their product is. Because the numbers are so small, we can do it easily. We could take a similarly small domain elliptic curve problem and, given things the same size, the nature of the elliptic curve problem is much more difficult than the nature of factorization at the same size. And what that means is we can use much smaller numbers with ECC to get the equivalent difficulty. And the fact that we're using small numbers means that when we're encrypting things, that's much faster.

So, for example, when web servers are wanting to use ECC to encrypt connections, they can do that with much less overhead than if they were using traditional RSA encryption, which because the keys are bigger, to do encryption, it just takes more processing in order to do it. So that's the whole tune-up on the relative difficulty of cracking different keys in the three different types of crypto.

Leo: I hope you're happy, Mary the IT Girl. I just hope you're happy. Now we move on to our final question, from Birmingham in the United Kingdom, a terrific TNO question from Mark: Steve, I love the show, long-time listener, proud owner of SpinRite, Password Haystack preacher, Vitamin D taker, SQRL evangelist, blah blah blah. I wonder if he wears a bearskin hat? But first-time question proposer. Steve, the company I work for, like many others, are often looking at outsourcing systems, and I'm intrigued to know more about Trust No One, TNO. I understand the concept of TNO, but I'm not sure I can apply it when auditing the security of outsourced services.

I understand that some of the telltale signs are if the outsourced service can do a password recovery. Is this true, and does it apply to recovery methods where you're not supplied with your original password, but are provided with a re-activation link? What else are the giveaways, the tells? And what direct questions would you ask vendors? How can we cut through the sales spiel and determine whether services are actually TNO or whether these suppliers also hold the keys to our data? If you get 'round to answering this question, it would be greatly appreciated. If not, I will not think any less of you because the work you do for the show is great. Much appreciated, the very polite Mark.

Steve: So this is a great question. You know, we've never had it posed before. I like the question.

Leo: TNO Tells.

Steve: So here would be the test. You would pretend to be Nervous Nellie, wanting to store your data in the cloud or with this purveyor. And you would say to them, innocently, with them not knowing that you were a listen of the Security Now! podcast for years...

Leo: No, because that's a real tell, yeah.

Steve: You'd say, okay, we're going to store our stuff with you. What if we lose everything? What if we...

Leo: I like it. What would you do? How would you help us?

Steve: Yes. How do we recover if the person that knows the password goes into a coma or is unavailable, and we have to access the data? How can you help us? If they can, they are not secure.

Leo: Right.

Steve: They are not TNO. The idea is, if it's possible for them to help you get your data back without you providing anything, then they can get it back with or without you by definition. And they could be giving your data to anybody who asked for it, under whatever terms and conditions. So that's the key. What you want to hear is them warning you. You want red flashing neon banners on their site saying, look, we'll store this for you. But you are entirely responsible for not losing control and keeping control of the access keys. You want them telling you, and believe me, they will tell you because they don't want the liability, if they can't recover it for you, they're going to make sure you know it's on you.

And if you want TNO, that's who you want. You want somebody who says, look, just be sure you understand the responsibility you're taking on here. We can't help you. We'll store your bits. But that's all they are to us is pseudorandom noise. It's on you to be able to decrypt them again. So if they instead say, oh, well, we have very good security, military grade, in fact, we actually - we have alien technology.

Leo: Navy Seals. Oh, okay. And Navy Seals.

Steve: Yeah, Navy Seals. No, the Navy Seals...

Leo: Alien Seals.

Steve: No, no. The Navy Seals negotiated with the aliens because you don't want just anybody to do your alien negotiations. So the Navy Seals did the alien negotiation to get their technology; and, thanks to that, we can recover your data if you lose everything, including all your access control. In that case, no TNO.

Leo: It's like that Washington Post article that said, oh, the solution to law enforcement's needs and our needs for privacy...

Steve: Oh, the golden key. The golden magical key.

Leo: ...is some magical, golden key that's held by unicorns.

Steve: Oh, yeah. Now, it's not a backdoor. We all agree...

Leo: No. No.

Steve: ...that there should not be a backdoor. But if Apple and Google could just come up with a golden key, then, you know, we won't call it a backdoor. It's the golden key.

Leo: It's the golden key.

Steve: Yeah, mm-hmm.

Leo: Yeah.

Steve: That would be nice.

Leo: Steve Gibson, GRC.com. You go there. You will find many wonderful things, including SpinRite, the world's finest hard drive maintenance and recovery utility; Steve's freebies - all the information you could ever want about SQRL, Vitamin D, Perfect Paper Passwords, Password Haystacks, everything's free except for SpinRite. SpinRite pays for it all.

Steve: Yup.

Leo: And it's a good place to go if you have questions. Not via email. Steve doesn't look at email. Just go the feedback form, which is at GRC.com/feedback. He also has 16Kb audio versions of the show. No one else has that. He has locked in the market. He has a monopoly on 16Kb audio for this show.

Steve: Yes, it's increasingly less important. But Elaine really likes it, although she's got a new satellite link. Apparently she's able to download things much faster, yeah.

Leo: Elaine takes that 16Kb audio, listens intently, and turns it into a typed transcript of the show, which we...

Steve: Often sends me notes in the middle of the night saying, what did this mean?

Leo: How do you spell that?

Steve: I listened to this three times. Sometimes I go - I'll listen to it, too. I have no idea what I just might have been saying.

Leo: Redacted. No, there's nothing redacted in her transcripts. But that's all at GRC.com. We have high-quality audio and video also at our site, TWiT.tv/sn. If you wish to subscribe, there are many, many podcast apps that will do the job because Security Now! is one of the oldest podcasts in the world and so has, as a result, kind of wormed its way into all the different podcast apps.

Steve: Now, when we're talking about Elaine doing the transcribing, and she's transcribing that we're talking about her doing the transcribing, isn't there, like, some sort of an infinite loop Mbius strip thing that happens? Like when you aim the camera at the monitor, and it goes down into infinity?

Leo: Yeah, it's the snake eating its own tail.

Steve: Yeah, yeah.

Leo: We do this show 11:00 a.m. - how was that, changing times, by the way? Did that work out all right last week?

Steve: Oh, yeah. Absolutely.

Leo: We're back to our regular time, though, 11:00 a.m. - I'm sorry, 1:00 p.m. Pacific time, every Tuesday, right after MacBreak Weekly. That's 4:00 p.m. Eastern, 2000 UTC. You can watch at TWiT.tv. And let's see, what else? I think that's pretty much everything.

Steve: Yeah, I'm completely bemused by Windows 10. The good news is apparently I don't have to worry about it for, like, until the spring sometime.

Leo: You know there's a big - and I'm surprised you didn't talk about this, but I'm going to. There's a big furor over it because in the license agreement for this beta software, not even beta...

Steve: This technical preview, yes.

Leo: They say we have the right to, not only watch everything you do, but we have a keystroke logger to monitor your keystrokes. That's normal. That's telemetry.

Instrumentation, sometimes they call it, that is commonly used for debugging purposes.

Steve: Especially when you might type something, and it collapses.

Leo: Right.

Steve: It's like, well, it'd be nice to know what you typed.

Leo: I'm going to take a wild guess here and say they probably won't put it in the release version of Windows 10. Now, that would be a problem.

Steve: Especially now that it's gotten as much press and attention as it has.

Leo: But, and then people say, oh, well, here's how you disable it. Don't disable it. You're making an agreement with Microsoft, if you want to use a technical preview, to help them. This is about beta testing.

Steve: Right.

Leo: That's what happens. You can't - you don't get to say, well, I'll beta test it, but you can't know anything I'm doing. That's not how it works. That's part of the agreement. You know I'm an old-timer now when I have to explain.

Steve: And besides, you know, Windows 10, at this point, apparently it's got a ways to go. Paul was saying they're going to be putting all kinds of more stuff in it. It's like, okay, what you have right now is a shell.

Leo: Yeah.

Steve: You know, it's like...

Leo: And try it, fine. But understand that you are now a beta tester, and there are rights and obligations that go along with that. And that does not include the right to turn off the logger.

Steve: Or the crash reporting technology.

Leo: Yeah, like that.

Steve: Which is the reason, the whole reason it's out there. They're spreading it around in order to make it crash so that they can go, oh, look.

Leo: Right. You're helping them.

Steve: Why did that happen, yes.

Leo: And if you don't like it, guess what, you don't have to install it. That's even easier.

Steve: Leo, you do have to, you know, they will give you your money back which you didn't pay.

Leo: Right. It's like when people complain about our shows. I'll be glad to give you a full refund.

Steve: Absolutely.

Leo: Steve, thank you so much. Always a pleasure. Lots of fun. We'll see you next week on Security Now!.

Steve: Thanks, my friend.