SECURITY NOW!

Transcript of Episode #627

Sharknado

Description: Although there are an unbelievable FIVE Sharknado movies, this will be the first and last time we use that title for a podcast! This week we have another update on Marcus Hutchins. We discuss the validity of WikiLeaks documents, the feasibility of rigorously proving software correctness, and the fact that nearly half a million people need to get their bodies' firmware updated. Another controversial CIA project is exposed by WikiLeaks. A careful analysis is done of the FCC's Title II Net Neutrality public comments. We talk about a neat two-factor auth tracking site, the Stupid Patent of the Month, an example of a vanity top-level domain, a bit of errata, and finish up with the utterly unconscionable security mistakes made by AT&T in their line of U-Verse routers.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-627.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-627-lq.mp3

SHOW TEASE: It's time for Security Now!, and it is a "SharknAT&To," this episode dedicated to a zero-day revealed by security researchers in every single U-Verse Arris modem, and it's a bad one. Lots to talk about. Also some interesting research which just came out by Brian Krebs on the Marcus Hutchins story. All is not as it appears, perhaps.

We apologize in advance. You'll notice during the show, and I hope it's not too bad, that there are audio breakups. We do know about it. Steve and I spent a long time trying to troubleshoot. I think we know what it is, but we weren't able to fix it for this episode. So my apologies for the occasional glitches and burps. Nothing we can do about it. We will not have this problem next week, though, I promise you. Stay tuned. Security Now! is next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 627, recorded Tuesday, September 5th, 2017: Sharknado?

It's time for Security Now!, the show where we protect your privacy and security online with this cat right here, this groovy individual named Steven "Tiberius" Gibson, the man at GRC.com.

Steve Gibson: With the extra ebullient Leo Laporte...

Leo: Yes, I am. I must have had a little something something.

Steve: ...as our copilot. I think you did, Leo.

Leo: Oh, wait a minute, I did. What was that weird drug you sent me? That thing that's supposed to improve your alertness? What was that, that weird vitamin? Wasn't PABA, but it was like it, that was...

Steve: Oh, I've forgotten now.

Leo: Yeah, remember that?

Steve: Oh. Hmm.

Leo: You need some, Steve. No, it's so funny because you said earlier that I was kind of peppy on Security Now!, I mean on MacBreak Weekly.

Steve: Oh, my goodness. Obviously so.

Leo: I didn't notice it. But it must be that thing. I only took 325 milligrams. Was it lysergic acid diethylamide?

Steve: No, that's not it.

Leo: Not it. It's probably...

Steve: I think I did give you a norepinephrine precursor; didn't I.

Leo: You did.

Steve: Yes.

Leo: Completely legal in most states, I might add. Wow. You know, I forgot I'd taken it. And that explains the peppiness.

Steve: Okay, yes. Back off from that, Leo. I think maybe...

Leo: I took a little dose, Steve. Okay.

Steve: Okay. So I have to apologize. I think this is the first podcast I've ever felt I had to apologize for the title. And although there are an unbelievable five Sharknado movies, this will be the first and last time we ever use that title for the podcast.

Leo: Well, you'd better have a good reason for it.

Steve: Oh, well. Whoever named this disclosure, it's like Honey Monkeys or Heartbleed. In this case it's Sharknado. It's the name of the set of vulnerabilities that were found in AT&T U-Verse modems. And they are, I mean, I was going to say heart-stopping, but that's a different story we have today. It's unbelievable. So the guys who wrote this up disclosed it immediately when they realized how bad this was, just to deliberately put pressure on AT&T because it's unconscionable what they have done.

Leo: I know what you're talking about. This is the problem with the Arris modems.

Steve: Yes.

Leo: Yes.

Steve: It's the Arris modems. And then AT&T OEMs them from Arris and then alters their own firmware. So there's some question as to - because it looks like it may be both. But when our listeners - I'm going to put it at the end of the show. We'll get to it. When our techie listeners understand how bad this is, in some cases in every single one of the AT&T U-Verse systems, for example, allows anyone on the Internet anywhere to access the ports on any of the machines behind the router. So essentially not requiring port mapping, but just kind of going straight through. It's just incredible. So I thought, okay, we have to further shine light on this. And it just is another perfect example of, boy, you know, doing it wrong.

So this is Podcast 627. We're going to talk about that. We've got another update, an interesting one, this time from Brian Krebs, just went live an hour or two ago. And so but it's an interesting take. Brian spent the last three weeks researching Marcus's background and drew, coincidentally, but he did the work, we were just guessing, exactly the same conclusion we had for the last few weeks about sort of the back story here. We're going to discuss the validity of WikiLeaks documents; the feasibility of rigorously approving software correctness. Nearly half a million people are getting a part of their body's firmware updated.

Leo: Ha ha, he said "firmware," ha ha. I'm sorry.

Steve: Calm down, Leo.

Leo: I'm sorry.

Steve: Don't ever take any more of that than you did.

Leo: It's DLPA.

Steve: Yup, DLPA. Yup, that's exactly right.

Leo: I've got to get a new source for this stuff because it's awesome.

Steve: DL-Phenylalanine is two forms of an amino acid that are precursors, yup.

Leo: Completely harmless. But better than coffee.

Steve: It's actually is very good for you.

Leo: Yes.

Steve: Anyway, another controversial CIA project exposed by WikiLeaks. A careful analysis of the FCC Title II Net Neutrality public comments comments.

Leo: Oy.

Steve: And it's really funny, too, because this was commissioned by the ISPs, and they did not like the results.

Leo: No.

Steve: A neat two-factor authentication tracking site. The Stupid Patent of the Month, which is actually the EFF's title for this, not even ours, being snarky. They were. And, boy, you won't even believe what that Patent Office has done this time. An example of a - we were talking about vanity top-level domains, like I could get .grc? And one just surfaced, and I thought, oh, that's just a perfect coincidence. A tiny bit of errata and a little bit of miscellany. And then we're going to discuss these utterly unconscionable security mistakes made by AT&T in their line of U-Verse modems.

Leo: And I have an apology to make because I know our audio, if you're listening now, and you hear it, is a little choppy. And we have no idea what's going on. We've been working on this, and we...

Steve: For 40 minutes.

Leo: Well, not just for 40 minutes, for months, because we sent you a new audio interface and...

Steve: Right.

Leo: ...we're trying to figure it out. But we'll keep working on it. I apologize. And we've decided that the content takes precedence over the quality in this case, and we wanted to get you that information. If you can't stand it, there will be handwritten transcriptions available at GRC.com in a week or so, or a few days, anyway. Do you want to take a break? I know, you've got to find some DLPA, man. This stuff is great. It's so funny because I did not - I forgot I took it this morning. And until you said "You seem unusually ebullient..."

Steve: Have other people noticed it? Has anyone...

Leo: No. Have you noticed it?

STAFF: You take it before you go to bed?

Leo: Well, I don't like to take it before I go to bed. It gives you nice dreams, elaborate dreams, but I like the energy that it gives me. Am I supposed to take it before I go to bed or before a show?

Steve: No, before a show.

Leo: That's what I say. Whoa. I'm going to calm down now.

Steve: So check out this Picture of the Week. It took me a while to figure out what it was I was seeing. This is the most embattled ATM I have ever seen. Oh, I mean...

Leo: Is this in the Caribbean? Where is this?

Steve: No, I can't - you know, the link did have the address, and I don't remember now where it was.

Leo: Maybe it's Venice Beach or somewhere. It's got...

Steve: Oh, my lord.

Leo: It's got, like, boards all around it. And then there's graffiti.

Steve: And that big red bar across is obviously to, like, keeping something from backing into it, or to keep you from ripping the ATM out and taking it away with you. I mean, it just - I just, like, first of all, one thought is why even bother having an ATM there? I mean, do you really have to have an ATM there?

Leo: Who would use it? Right? Don't you...

Steve: Exactly. It looks like you'd get a disease just walking by.

Leo: It's like the old phone booths. That's what phone booths ended up looking like. Holy cow.

Steve: Oh, yikes. Anyway, I just got a kick. It was like, here is technology trying to struggle against all odds. And, you know, at this point skimming, you know, sticking your card in, having a skimmer is the last of your worries. I mean, I just - I wouldn't even get near that thing. It looks like it would bite you.

So Brian Krebs, bless his heart, spent the last three weeks digging deep. And this is, you know, the moment I saw this I thought, oh, this is perfect for Brian. This is what he does. As we know, he's interested in going back and really doing, like, the dark corners of the 'Net, taking the time to create identities that are not linked to him so that he can participate in the dark underbelly chatrooms and so forth. And then he brings his results to us. So I was just, as I said at the top of the show, just made aware of his most recent posting, which I've linked to here for anyone who wants to dig in. And I haven't had a chance. I just, like, I understood what it was he said. So I've not been able to dig in. I will go through it before next week to see if there's anything more worth sharing. But to quote just the headline summary, here's what he said.

He said: "At first" - I'm sorry. I forgot. We're talking about Marcus Hutchins, who as we know was arrested by law enforcement as he was trying after DEF CON to go back to his home in the U.K., and he was detained. He's now, speaking of Venice Beach, Leo, he's now in Venice Beach with two attorneys, waiting for trial, I think next month, in October.

So Brian said: "At first, I did not believe the charges against Hutchins would hold up under scrutiny. But as I began to dig deeper into the history tied to dozens of hacker forum pseudonyms, email addresses, and domains he apparently used over the last decade, a very different picture began to emerge. In this post," writes Brian, "I will attempt to describe and illustrate more than three weeks' worth of connecting the dots from what appear to be Hutchins' earliest hacker forum accounts to his real-life identity. The clues suggest that Hutchins began developing and selling malware in his mid-teens only to later develop a change of heart and earnestly endeavor to leave that part of his life squarely in the rearview mirror."

Leo: That kind of makes sense, yeah.

Steve: It does. It makes total sense. So yay to Brian for doing that work. Thank you, Brian. And of course it exactly corresponds with our own, what was just gut feeling. Like, well, you know, all the evidence was he wasn't happy that his recent work was being abused, but there had to be something more. And I'm just hoping - I don't know what -

whose laws apply where, if there are exceptions for underage, that is, non-adults. But maybe that will give him some shield. If he's kept his act clean and been a good guy as an adult, maybe his earlier teen antics are shielded by not being at the age of consent and so forth. So I don't know. But anyway, I just wanted to provide Brian's really useful research feedback. Again, thank you.

Okay. Somebody following up on our comments last week about needing to remember - and you highlighted this for us, Leo. We don't know if WikiLeaks document dumps, which are so damaging to law enforcement actual technology use and reputation, are real because we just don't know. But someone sent me a link which initially looked like there was a way to prove it. It turns out that's email being proved, not documents secreted somehow from the CIA.

So one of our listeners responded to my caution about placing possible unwarranted trust in WikiLeaks documents with a link. Now, this is an older posting from about 10 months ago, October 2016, and by somebody with a lot of cred, Robert Graham, who's the guy at Errata Security. And of course he's our old buddy fro the BlackICE personal IDS firewall days, in the early days of Windows firewalls. Theirs was a different kind of product that's called BlackICE that was more of an IDS. It was more of an intrusion detection system for a PC than just a simple firewall. And so it wasn't rule-based, and it wasn't my kind of solution. I just believed you should close the ports down and open them wherever you wanted. Theirs was sort of more automatic, but a useful product.

Anyway, WikiLeaks documents are not leaked emails. So Robert's blog posting is only relevant to email, but due to a technology originally developed as an antispam solution, as a direct consequence incorporate cryptographic signatures that, as cryptographic signatures will, can be used to authenticate a document and prove with cryptographic certainty that it has not been altered. So we can validate WikiLeaks emails, or some WikiLeaks emails.

Okay. So before I go any further, I want to remind everyone, because this is a political case, that this is not a political podcast, and I work scrupulously to keep us from falling into that controversy. I don't know what Robert's politics are, even if he is political, but this isn't about that, though it touches on something from our last election cycle.

Robert writes in his posting last October, October 2016, he said: "Recently, WikiLeaks has released emails from Democrats. Many have repeatedly claimed that some of these emails are fake or have been modified, that there's no way to validate each and every one of them as being true." He writes, "Actually, there is, using a mechanism called DKIM." That's DomainKeys Identified Mail. "DKIM is a system," he writes, "designed to stop spam. It works by verifying the sender of the email. Moreover, as a side effect, it verifies that the email has not been altered."

He goes on: "Hillary's team uses HillaryClinton.com, which is DKIM-enabled. Thus," he writes, "we can verify whether some of these emails are authentic." Now, there's a caveat there that I'll get to in a second. He said: "Recently, in response to a leaked email suggesting Donna Brazile gave Hillary's team early access to debate questions, she defended herself by suggesting the email had been 'doctored' or 'falsified.'" And he writes: "That's not true. We can use DKIM to verify it."

Then he provides a link to the specific email and notes: "It's not a smoking gun; but, at the same time, it both claims they got questions in advance while having a question in advance." And then he said: "Trump gets hung on similar chains of evidence, so it's not something we can easily ignore." I don't understand what he meant by that, but regardless.

Anyway, this post isn't about the controversy - thank you, Robert - but the fact that we can validate email. When an email server sends a message, that is, a DKIM-enabled email server, it'll include an invisible header. They aren't, you know, email headers aren't specifically hidden. Most email programs allow you to view them. It's just the way, you know, there's a bunch of debris that confuses people, so we don't want to see them.

And then he goes on to explain that normally this email validation is server to server, that is, from SMTP server to [dropout], meaning that when a sender of email sends their email from their POP or IMAP client to an SMTP server, then it's got a private key. It takes a hash of the entire email - subject line, time and date, body of the email. It hashes that, and then it digitally signs the hash to create a signature which cannot be forged. Then [dropout] to its destination. The receiving SMTP server receives it, sees that it has a DKIM header, meaning that this particular piece of email is verifiable, it is validatable affirmatively.

So if it doesn't already have the private key in its cache of candidate recipients - and, you know, there are so many domains in the world it probably doesn't - it does a DNS lookup. And this is one of the reasons, you know, our listeners know that I'm so excited about the idea of DNS being this generically useful database, an Internet-wide cacheable index. And thus when DNSSEC finally happens, to be able to have that robustly secured is going to create an incredible asset for the world.

So in this case there's a text record which is an additional record for a domain. So, for example, HillaryClinton.com had a text record which allowed that domain to publish its public key. So it signs outgoing email with its private key, which it keeps secret and in control. But then any recipient can go, oh, let's make sure this really did come from HillaryClinton.com, and not a single character was altered. In that case, the receiving SMTP server performs a special DNS query, not asking for an IP address, but asking for the text records which are just textual that are associated with that domain name. That allows it to get a base64 encoded version of the binary public key. It decodes that into binary, then it uses that public key to verify the signature. So all of that happened.

Well, it turns out there's an extension which you can get. He uses Thunderbird. So he said: "How do you verify this is true? There are a zillion ways with various DKIM verifiers." And he writes: "I use the popular Thunderbird email reader from the Mozilla Firefox team. They have an add-on designed specifically to verify DKIM." Meaning that in this case your email client itself does that same verification. It queries the remote DNS, gets the public key, and shows you as you're looking at email whether it's verified or not. So it's kind of a cool add-on for an email reader. I mean, it doesn't get in your way at all. It'll let you know if this is spam or not. If it's DKIM-signed, the chances are way lower. At least it's not spoofed. You could have a DKIM-signed spam email, but it can't also be a spoofed domain. So once you decide you don't want any more email from that server, you can put it on a blacklist, and there's no way it could jump around.

So anyway, he says: "Downloading the raw email from WikiLeaks and opening in Thunderbird with the add-on, I get the following verification that the email is valid." And he provides us with a screenshot. He says: "Specifically, it validates that the HillaryClinton.com sent precisely this content, with this subject, on that date."

So that's all true. The problem is this evidence appears to be very damning, but this is - we have here a security and technology podcast, so we always ask the question, how sure are we? Cryptographically, we're absolutely as positive as we could be that the signature is correct. However, as we've often discussed in the case of HTTPS connections with TLS certificates, and this applies [dropout] with DKIM, the certificates, that is, the private keys and the security of the system absolutely relies on the assumption that the signing server is uncompromised because it's the signing server that has the private key that allows it to perform this authentication or to provide the authenticatable content.

But the whole point of this, in this case the Hillary email debacle, was that their email server was compromised. And as we know, if in addition to the 3,000-some emails that were reportedly obtained, if that private DKIM key was also obtained, then we're right back where we started, with a somewhat harder to substantiate claim of document doctoring, but still a way that it could have happened. So this stuff, we get down in the weeds with the technology. But we've never really talked about DKIM in the entire 12-plus years of the podcast, so [dropout].

And this doesn't, of course, speak to WikiLeaks documents and their forgeability or not, only to email, and only to email that is DKIM-signed, and only in that case where you can reasonably assume that the DKIM signature could not have been spoofed. But unfortunately, in this case, where a server compromise was part of the story, you really can't make that assumption. So [dropout] we did before in terms of secure crypto-level assurance, which we would like to be able to have in this case, but the facts really don't support that. But still, an interesting twist.

We've talked some time ago about this questionable deterministic random bit generator which used the dual elliptic curve technology which the NIST had standardized on against all reason. There are four that they have in their kit. And this is - and I don't have the - oh, yes. It's NIST document 800-90A. [Dropout] hash-based version, an HMAC-based version, meaning that it's a keyed hash. There's a counter running through a cipher based version. And then of course the infamous dual elliptic curve DRBG, which is based on the elliptic curve cryptography. And remember that the problem with it that surfaced a couple years ago was that there was no - the NSA provided the curve parameters with no substantiation for where they came from. They just said, "Here are these magic numbers."

Well, the problem is those magic numbers could have been hiding a backdoor, which could have allowed someone who had some of the random numbers that were being produced to get the other ones. And then, to make matters worse, it was the slowest of the four. There's, like, four different algorithms, and the other three are obviously strong. But, for example, RSA chose the slowest one, that one. So it was like, again, we had no proof of anything, but it just seemed like a compounding of suspicious circumstances.

So all of this by way of the fact that Matt Green and a team went to the trouble of absolutely verifying everything about one of these. They took the HMAC-DRBG and have published a paper out of Princeton, verified correctness and security of mbedTLS, that's M-B-E-D-T-L-S, which is an embedded TLS crypto that happens to use this HMAC deterministic random bit generator to generate its pseudorandom numbers.

Okay. So I'm going to explain this a little bit, but I want to talk more - what I really wanted to do was to use this as a jumping off point to talk about software verification. So they write: "We have formalized the functional specification of HMAC-DRBG, and we have proved its cryptographic security, that its output is pseudorandom" - which sounds like not a big deal, but that's a high bar. And when these guys say "prove," they're not saying "we're of the opinion," or "five of us all looked at the code." They're saying, no, "We have proved it," they said, "using a hybrid game-based proof. We have also proved that the mbedTLS implementation written in C correctly implements this functional specification." And of course that's exactly what we want. We want proof that both the theory and the practice, which is to say in this case the implementation, are both verifiable, both solid.

So they continue: "That proof composes with an existing C compiler correctness proof to guarantee, end-to-end, that the compiler's output machine language program gives strong pseudorandomness." They write: "All proofs - hybrid games, C program verification, compiler, and their composition - are machine-checked with the Coq proof assistant." So there's some automated assistance to help unwind any spaghetti in a code and absolutely nail down what's going on. They said: "Our proofs are modular. The hybrid game proof holds on any implementation of HMAC-DRBG that satisfies our functional specification. Therefore, our functional specification can serve as a high-assurance reference."

And so what I wanted to bring to this is that this is the future. We're not there yet. But computer programs are just math. They're not a crapshoot. They are a deterministic processor. Assuming there are no bugs in the processor, so we have had them in the past, so we have to make that caveat. But if the processor properly follows the instructions, each instruction does exactly what it's supposed to. And then there are other problems like cache conflicts or instructions interfering with each other. So it gets complicated quickly.

But computer programs are not flowers and trees and fields of stars. They are absolutely and utterly subject to pure and perfect rigorous analysis and verification. The problem is it's difficult. It's unclear how fast we're going to see progress in the area. But this is a larger chunk of verification than I remember seeing recently. So it feels like we're getting better at it. I don't know if it'll happen in our lifetimes. On the other hand, we're seeing exponential growth in AI capability, so that might be a factor. Ask Watson if the program is provably correct.

And this is why, for me, as a writer of code myself, always chafed at the lay statement which unfortunately is true, that all software contains bugs, which we've all heard that said before. We may have said it ourselves. I would back off from that, though, and say that trivial software can be easily bug-free. However, more complex software, for example, like the U.S. Shuttle's flight guidance and navigation computer, which needed to be bug-free, can be bug-free, if it's sufficiently important to then pour incredibly extensive resources into it in order to make it so.

So my point is that it's the nature of software to get rapidly insanely complex as it grows in size. The interactions within a program just become crazy. So very quickly a trivial small program, like a loop that adds numbers. Okay, I want to compute a Fibonacci. Okay, that's three lines of code. It has no bugs. I mean, it's so simple, it's clearly perfect. It's provably correct. But you start adding bells and whistles to it, and things go out of control quickly. And that's the point. The complexity just exponentiates. It goes insane, and so out of our ability to perfectly know that it was exactly correctly implemented.

So I think what's destined to happen - and again, I have no sense for timeline, but we need it to happen. We are getting the capability for it to happen, and so I think it will - is that what will happen over time as the computing resources increase, and they're clearly going in that direction, is that the cost of rigorous software correctness proofs and verification will fall dramatically so that we're no longer guessing that it's okay because it stops not being okay, and so we go, whew, ship it. Instead, we tell Watson, okay, here we go, does this do what we think it's going to do?

And, I mean, this is not a small problem to solve, which is why it has so far been virtually intractable, and it is so insanely expensive because of the fact that we just don't know how to do it yet, that none of our software, I mean, virtually none of it has been proven to be correct. We kind of limp along and get updates and patches and say, well, okay, because that's a lot cheaper. But imagine a day where we have AI that is able to do what we cannot do, which is to find the bugs that are there, that are not manifest, and say, whoops, you meant to do this over here, didn't you. Oh, yeah, thank you for telling me now.

So anyway, so Matt and his group did that. They wrestled a substantial piece of work to the ground, producing mathematical certainty that it is correct, which is very cool. And you and I, Leo, may not live to see the day. But in the future you could imagine that, because it is just math, because these are deterministic systems, they could be, given sufficient focus, made perfect, which would be nice.

Leo: Wow. Yes, for those of you watching at home, Steve has lost the caterpillar that used to haunt his face. We commented on it last week, but just in case people didn't tune in last week, or didn't watch the video last week, he knows his moustache has gone missing.

Steve: Yes. And it requires maintenance to keep it so.

Leo: Yeah. Hey, I got a Dollar Shave Club for you. I'll send it out. It's right here somewhere. Yup.

Steve: So when I saw this title I thought, wait a minute, how do you do that? The FDA is recalling 465,000 pacemakers.

Leo: Uh-oh.

Steve: But that's the definition - that's an embedded computer.

Leo: Yeah, literally. In your chest cavity.

Steve: And so this story will make your chest thump. The United States FDA, the Food and Drug Administration, is recalling 465,000 pacemakers to which attackers - get this -- attackers can gain unauthorized access - yes, to your pacemaker - to issue commands, to change settings, and maliciously disrupt its function.

Leo: Didn't we talk about that hack some months ago? It feels like we talked about that hack.

Steve: It's probably coming back around again, yeah. Oh, in fact, this is not the first time that Abbott Labs has had a problem. You're exactly right, Leo. There was a related problem that they had a few months ago, so good memory. According to the FDA, the recalls of affected pacemakers are tied to research by MedSec Holdings. I guess they were the researchers that originally brought the St. Jude medical equipment flaws to light about a year ago, and Abbott Labs acquired St. Jude Medical in January.

Okay. So I snipped a bunch of nonsense out of the reporting because the details of the exploits are chilling. And I didn't even realize, we've talked about US-CERT, the Cyber Emergency Response Team? Turns out there's an ICS-CERT which is the Industrial Control System Cyber Emergency. And apparently that's the category for pacemakers. You've got an industrial control system in your chest, implanted or embedded. So the ICS-CERT manages industrial control system vulnerabilities, catalogs them and tracks them and so forth. They cite three vulnerabilities in their advisory regarding Abbott Laboratory's pacemakers, which were manufactured prior to August 2017, meaning last month. So unless your scar is really fresh, you may have one of these puppies.

The highest rated of the three vulnerabilities is [dropout] authentic - it's hard to even say this - the pacemaker's authentication algorithm, the authentication key and timestamp can be compromised or bypassed. So there's an authentication bypass on your pacemaker which could allow a nearby attacker, because this is RF, so it's radio, so someone has to get in your proximity, to issue unauthorized commands to the pacemaker over the air, wirelessly. An additional bug could significantly reduce battery life of the pacemaker. You would like your implanted pacemaker's battery to get the full benefit of its life so that you're not having to be reopened again.

The CERT advisory said the pacemakers do not restrict or limit the number of correctly formatted RF wakeup commands that can be received, which may allow a nearby attacker to repeatedly send commands to reduce the pacemaker's battery life. So in other words, as we know, receiving radio is a low power task compared to sending it. And this is a bidirectional interface between the outside world and the pacemaker in a person's chest. So the pacemaker is either always or periodically receiving, but that is very low power, doesn't take much to receive. But when it's explicitly awoken, then that fires up the transmitter, and transmitting energy is far more power consuming than receiving it.

So the second half, so [dropout] problem with the firmware is that there is no limit on an attacker's ability to drain the battery of an affected pacemaker by keeping its transmission radio running all the time. So you can imagine an attack scenario where an office worker who's generally going to be in a certain location is having his pacemaker drained without his knowledge, and certainly that's not what he wants.

And then, finally, the third flaw is related to the fact that the devices - again, really? The devices transmit unencrypted patient information via RF communications to programmers and home monitoring units. So they didn't bother to run a cipher on this. It's just plaintext coming out, which is problematic, they write, oh, because the pacemakers store patient data in the clear and transmit it in the clear. So that just feels like sloppiness. I mean, you could say, well, some things are a mistake.

But as we've discussed often, it's important to separate policy from error, and it's hard to defend a person's pacemaker transmitting plaintext to any receiver within range, which apparently these things have some range. If you've got a home monitoring unit, maybe you can walk around while it's monitoring you. So anyway, they have been recalled, 465,000 pacemakers. Now, the good news is they are firmware-upgradeable by that same radio link. So we're not talking about any more open hearts or open cavity surgery. It's just, okay, fine, we need to see you in the doctor's office to update your body's firmware.

"The vulnerabilities could be exploited," the advisory writes, "via an adjacent network. Exploitability is dependent on an attacker being sufficiently close to the target pacemaker as to allow RF communications." I'm not talking - I'm sure it's not Ethernet commercial WiFi. You know, it's going to be some wacky medical band RF. So it's not like your neighbor is going to receive it on their television set or something. It's definitely in its own world of RF spectrum. But if anybody were targeting someone, that would not represent any kind of a problem because we also have now very inexpensive SDRs, Software-Defined Radios, that you just set to whatever you want.

So anyway, mitigation of these problems require patients to visit their doctor for a short-range wireless update. Abbott warns...

Leo: It's cool that they can do that, though, I have to say.

Steve: Yes, yes.

Leo: And it's a relief, too.

Steve: And I'm sure that this is all super regulated so the devices are serialized, and I'm sure there's a database so that they're able to get notices to the - I was going to say the "end user" - but to their doctor, you know, and follow up on this to bring everybody back in and get it taken care of. But yes, Leo, I agree. The idea that you can - I'm sure they just put something on your chest, and it has a dialogue back and forth. Abbott did warn that the firmware updates should be approached with caution, that is, they really don't want anyone messing with an already in-place pacemaker. They said: "Like any software update" - well, we don't want to think that this is like any software update, but that's what they said - "firmware updates can cause devices to malfunction." Gulp.

Leo: That wouldn't be good.

Steve: Gulp. "A botched update could result in a loss of settings to complete loss of device functionality."

Leo: Uh-oh.

Steve: So, now, the good news is you will be in your doctor's office. And not all pacemakers are, like, replacing a bad sinus node which has stopped functioning. Some of them are just like demand pacers, where if you had a little bit of a systole, it notices it and goes whoops and gives you a pacing spike. So it's not like stopping would immediately kill you, but it would be good that you've got your doctor standing by with some paddles. So, interesting.

Leo: Paddles. I hope that's not required. We'd better book some surgery this minute, yeah. Wow.

Steve: Okay. So here we have a case of another WikiLeaks dump where, boy, the press got this wrong. The headline is "CIA Caught Planting Malicious Software in Windows." Okay. But, no, that's not what happened. And the codename, as all these things have fun codenames, is AngelFire. So the truth is that a team of hackers at the U.S. CIA, our Central Intelligence Agency, allegedly used - again, from WikiLeaks, so maybe - used a

Windows hacking tool against its targets to gain persistent remote access.

Leo: So what's the surprise here? I mean, of course they did.

Steve: Exactly. Exactly.

Leo: What would you have them do?

Steve: And I'm glad they can do that. I'm glad we have...

Leo: Yes, break into the house? Put a glass against the wall?

Steve: Yeah. So as part of its Vault 7 leaks, WikiLeaks last Thursday revealed details of a new implant developed by the CIA named AngelFire to target computers running Windows. The AngelFire framework implants a persistent backdoor on target Windows computers by modifying their partition boot sector. In other words, it's a high-end, well-designed rootkit, specifically for intelligence and law enforcement data gathering. It's got five components. I mean, it's a beautiful piece of work.

AngelFire is the umbrella. There is something called Solartime, which is the thing that modifies the partition boot sector to load and execute the Wolfcreek, which is the kernel code, every time at boot up. So that makes it persistent over the long term. Wolfcreek, in turn, is a self-loading driver that runs in the kernel, which loads other drivers and user-mode apps. Keystone is the component that utilizes DLL injection technology to execute the "malicious" user applications. And again, "malicious" should be in quotes because it's intended for this purpose. They execute the purpose-specific applications directly into the system memory without needing to drop them in the file system, so there's no sign of them.

There's something called BadMFS which is a covert file system which attempts to install itself in non-partitioned space available on the target computer and stores all drivers and implants. And we've seen rootkits do this. I remember talking a long time ago that, as a consequence of the weird history of hard drives, there is this cylinder alignment. Cylinders don't exist anymore, like logically, the way we see them. We used to have cylinders, heads, and sectors. And partitions used to have to end. They didn't have to start on a cylinder boundary, but they had to end on one, meaning that if the drive - and a cylinder, especially when you have lots of heads or virtual heads and sectors, a single cylinder could be - I used to know this by heart, but it's number of heads times number of sectors times 512. That many bytes, a big chunk of space.

So the idea would be a hard drive itself, which is now actually just a linear array of bytes, it's not going to be an even multiple of the OS's arbitrary cylinder size. So there will be slack at the end. And in fact in 6.0 of SpinRite I understand that, and I specifically test all of the slack space and the intercylinder space by rounding up and down appropriately and then clipping to size so that SpinRite doesn't miss anything. But the OS does. And so these guys are using something that will be completely invisible. It's on the drive, but you can't see it. You can't get to it.

And then the last bit - those were the first four. The last piece is the Windows Transitory

File System, which is a new method of installing AngelFire that is a method these guys came up with which allows the CIA operator to create transient files for specific tasks like adding/removing files to and from the AngelFire system. So there's a user manual for it that WikiLeaks downloaded. The 32-bit version of the implant works against Windows XP and Windows 7, while the 64-bit implant can run on Server 2008 R2 and Windows 7.

So as so many of these things are, it feels a little bit dated, okay, so like 7, but not 8.1 and 10. But, you know, it's very likely that they adapted AngelFire 2 or 3 or something if they wanted to use this. But once again we've seen overblown and clickbait coverage of this in the press. For example, I read one report which opened with, quote, get this, I mean, I was worried when I read this because this was the first thing I encountered. I thought, what? It says: "All Windows machines have been infiltrated by the CIA under a project..."

Leo: Well, thank goodness. I think maybe they mean all Windows versions or - I don't know. That's weird.

Steve: Yeah, but the headline [dropout]: "All Windows machines have been infiltrated." It's like, no, they haven't. And then it's "allowing the U.S. government to load malicious programs onto a person's computer without their knowledge." Yes, a foreign adversary's computer. I mean, no one suggests that this should ever be perpetrated against U.S. citizens in contravention of U.S. law and the U.S. Constitution. And no one is suggesting that they ever did. This is very likely part of their toolkit for remote intelligence gathering. I mean, and we don't have any evidence one way or the other. But that's the assumption. And so I'm glad they've got power tools, you know, as long as they point them somewhere else. And I presume they do.

Okay. So in a little bit of turnabout is fair play, a consortium of ISPs commissioned an independent third party, because they wanted clean hands, to analyze all of the submissions which were made on the topic of Net Neutrality and Ajit Pai's threatened or planned anti-Title II plan. Ars Technica has a nice report about this, falling under the category of "be careful what you ask for."

Ars wrote that: "A study funded by Internet service providers has found something that Internet service providers really don't like. The overwhelming majority of people who wrote unique comments to the Federal Communications Commission want the FCC to keep its current Net Neutrality rules and classification of ISPs as common carriers under Title II of the U.S. Communications Act, according to the study which was just released." I've got a link to the study, for anyone who's interested, in the show notes.

And get this: 98.5% of the unique Net Neutrality comments said don't change anything. The study was conducted by a consulting firm Emprata [E-M-P-R-A-T-A] and was funded by the so-called Broadband for America coalition whose members include AT&T; CenturyLink; Charter; CTIA - The Wireless Association; Comcast; NCTA - The Internet and Television Association; the Telecommunications Industry Association (TIA). Oh, and USTelecom. So, you know, a lot of big players.

When Emprata analyzed all 21.8 million comments, including spam and form letters, 60% - if you just looked at them all, without discrimination, 60% were against FCC Chairman Ajit Pai's plan to repeal the Title II classification, and 39% supported the repeal. But the numbers shifted starkly, Ars writes, in favor of keeping the Title II rules when excluding duplicates in order to analyze just unique comments written by real people.

Emprata wrote: "There are considerably more 'personalized' comments, appearing only once in the docket, against repeal," meaning in favor of leaving everything alone. That is, 1.52 million were "leave it alone" versus only 23,000 personalized, individual, probably real comments supporting repeal. "Presumably," Emprata says, "these comments originated from individuals that took the time to type a personalized comment. Although these comments represent less than 10 percent of the total, this is a notable difference." Meaning, if you look at them all, it's a 60/40. If you look at ones that are probably authentic, it's a 98.5.

Leo: Yeah. I wrote a personal long comment, as long as I could fit in there.

Steve: Yes.

Leo: Saying why Net Neutrality needs to be preserved for my personal business.

Steve: Yeah. So I don't know if it'll make a difference. I just wanted to report on this interesting step because we've talked several times now about this flood of spam that, I mean, and DDoS attacks that basically just knocked it off. They had to take it down off the 'Net for a while in order to let things cool off, and then they put it back on again. So again, unfortunately we're in a world where, as we know, the guys with the big bucks can pay the big lobbying firms to apply strong pressure on our politicians to do what they want to have done. So we'll keep our fingers crossed.

Leo, you're going to love this one, a fabulous site: TwoFactorAuth.org, T-W-O-F-A-C-T-O-R-A-U-T-H dot org. I recommend it without hesitation to our listeners: TwoFactorAuth.org. The URL is in the show notes. It is a comprehensive listing. Now, Leo, click one of those icons. What happens is it opens the row below and gives you a breakdown of the type of authentication offered by a huge range of services and sites within the category that the icon uses. So there's communications, there's cloud providers, there's all kinds of different categories. But it shows you, for example, that row on the far right is what we want. That's the software time-based solution.

Leo: So that's Authenticator or something like it, Google Authenticator.

Steve: Exactly, exactly. The one in the middle is what we're now sketchy about, and that's the SMS authentication, where they're sending it in real-time. But this is just a cool reference for, like, I can imagine somebody really security conscious, when shopping for a service, a bank or a cloud provider or whatever, might say, well, I want one with time-based token two-factor authentication. This lets you shop the industry for that. And you'll see the links there.

They also provide you, in instances where a provider or a service has none, you've got links to make it easy to tweet your request that they add it to help apply pressure for second-factor authentication where it doesn't exist. So TwoFactorAuth, spelled out, dot org. And I commend it to our listeners. Just take a time and scroll around. It's well organized, and it's got GitHub behind it, and it accepts user submissions of things they don't have yet, and it's constantly being maintained. So it's a really...

Leo: It's a GitHub repo, so that's an interesting way of keeping it up to date.

Steve: Right.

Leo: So you can submit - which means that people are continually keeping it up to date. I was curious how old it is. Last commit was a day ago. So people are - and last commit was Stripe added UTF support. So you can add a commit, if you know enough how to use GitHub, when something changes, to do it. Which is great. Which is really great, yeah. And in fact you could install your own local copy of it, if you wanted. I'm not sure why you'd want to, but because it's GitHub you could do that.

Steve: Nice.

Leo: You could run it locally.

Steve: Well, yeah. I mean, maybe - you're right. I guess you typically wouldn't want this offline because you're inherently - it's an online process. But, yeah, to make a snapshot, that'd be cool.

Leo: I'm seeing more and more of this kind of collaborative stuff. Instead of using a wiki or some other solution, using GitHub and git commands to keep it up to date. It's kind of a cool way to do it.

Steve: It's a cool application, yeah. So the EFF labeled this in their posting, as in their URL, the Stupid Patent of the Month.

Leo: Oh, boy.

Steve: And, Leo, JPMorgan got a patent, I'm not kidding you [dropout], just make sure you're centered, on interapp permissions.

Leo: Oh. I'm glad they invented that finally.

Steve: Oh, god. Somebody had to, Leo. It was just staring us in the face. Why didn't anyone think of that before?

Leo: A system and method for communication among mobile applications.

Steve: Groundbreaking. Groundbreaking.

Leo: Groundbreaking.

Steve: Shocking. So the EFF writes: "We have often criticized the Patent Office for issuing broad software patents that cover obvious processes. Instead of promoting innovation in software, the patent system places," they write, "landmines for developers who wish to use basic and fundamental [and, by the way, old] tools."

Leo: Well, as I remember, this is how a modem handshake works. They've patented [mimicking modem sounds].

Steve: Yes, the sound of geese mating. "This month's stupid patent," they write, "which covers user permissions for mobile applications, is a classic example. On August 29," so last Thursday, "the U.S. Patent Office issued Patent 9,747,468" - which they now refer to as the "468 patent" - "to JPMorgan Chase Bank," the geniuses there who came up with this patent "titled 'System and Method for Communication Among Mobile Applications.'" It's a breakthrough, Leo. Who ever heard of mobile applications communicating with each other?

"The patent covers the simple idea," they write, "of a user giving a mobile application permission to communicate with another application. This idea was obvious," they write, "when JPMorgan applied for the patent in June of 2013." So only four years ago. How old is the iPhone? It had the little thing that popped up and said, you know, do you want this app to have access to your photos?

Leo: Right, right.

Steve: "Even worse," they write, "it had already been implemented by numerous mobile applications." And built into iOS. "The Patent Office" - I'm surprised that Apple didn't get a patent on it, but anyway. They probably knew they didn't invent it, either. "The Patent Office handed out a broad software monopoly, while ignoring both common sense and the real world." They provide in this the full text...

[Leo playing modem sounds in background]

Steve: Yup. They provide the full text of Claim 1 of the 468 patent. There are just five clauses, so I'll read them: "A method for a first" - and this is like legal speak patent talk - "A method for a first mobile application and a second mobile application on a mobile device to share information, comprising the first mobile application executed by a computer processor on a mobile device determining that the second mobile application is present on the mobile device; receiving, from a user, permission for the first mobile application to access data from the second mobile application; the first mobile application executed by the computer processor requesting data from the second mobile application; and, finally, the first mobile application receiving the requested data from the second mobile application." Again, Leo, shockingly brilliant. It's like, wow.

Leo: How did they think of that?

Steve: That's it. "The claim," they write, "simply covers having an app check to see if another app is on the phone, getting the users' permission to access data from the second app, then accessing the data." Anyway, and, see, for those who - we've talked about patent trolls and the East Texas nightmare and all that, Samsung putting their logos and supporting the local sports team, trying to beg for, you know, trying to say we're not evil to this wacky judge that just immediately, you know, who knows what kind of house he's living in and where the money is coming from. But it just seems all so slimy. The problem is, what JPMorgan now has is a license to sue. And you faced this with the podcast patent, Leo.

Leo: Yup.

Steve: It's like, so suddenly someone can strong-arm small players who face the expense of defending themselves against a specious claim in patent court, in patent litigation. And the way our system works, because JPMorgan Chase in this instance has the patent, you can't call this a malicious prosecution. So you're not able to sue for damages. They have a right to sue somebody who's infringing the patent they received legitimately from the U.S. Patent and Trademark Office. So it's just - it's a disaster when this happens.

And so the problem is this will end up being overturned. But a huge amount of cash is going to flow back and forth within the system to attorneys and patent attorneys and courts and dockets and everything, in order for a secondary process to kill something at great expense and time and emotional anguish on the part of the people who are targets of the patent holder, only to undo what should have never been done in the first place. And I know that it's a rough job. But unfortunately, patents are now being abused like this.

And what happens, of course, is that so many large companies have "patent portfolios," as they call them. And we've seen the Apple thing. I mean, Apple's patenting the sunrise in order to have it so that then they can have a cross-license agreement with IBM and Microsoft and Google so that they'll all basically agree to be able to not sue each other over the contents of their respective patent portfolios. But again, the system is just broken in a way that is just so demoralizing and sad.

Okay. And finally, before we take our last break, we talked - you and I were talking, and you surprised me last week by saying that, oh, yeah, anybody can get a top-level domain, a vanity top-level domain. You know, I could get...

Leo: If you've got the money, yeah.

Steve: If you've got the money. Oh, and the thing that occurred to me after we talked about it, Leo, is I'll bet that's not just a one-time fee. I'm sure you're paying every year for this thing.

Leo: Possibly, yeah, yeah.

Steve: In order to maintain it. And actually suggesting that is the fact that, believe it or not, the .montblanc domain...

Leo: Wow.

Steve: Montblanc had their own TLD. The IANA had a - I have a link to it in the page. Essentially somebody commenting on it said: "We rarely see entire top-level domains killed." Because of course typically you've got so many second-level machines in there. Like, you know, .com is never going to go away. It would kill the world. So Montblanc, someone thought, oh, that'll be fun. Let's grab our own top-level domain. It was removed last Friday. So not only did they apparently no longer want it, but they must have no longer decided it was worth the money that they were having to pay in order to hold it. So I just - this was just a coincidence that this came along. I thought that was kind of cool.

Leo: I wonder if it's the Montblanc pen company? Must be.

Steve: That's immediately where my mind went.

Leo: Yeah. Unless it's Switzerland.

Steve: So you can't go there now because it - yes.

Leo: Maybe Switzerland registered it.

Steve: Yow. Last break.

Leo: Nope. There's no more ads. We've run out of ads.

Steve: I forgot. You and I were so...

Leo: We didn't talk about it, yeah, no.

Steve: ...tied up with the quality of the connection. Okay, cool.

Leo: I know, I'm sorry to disappoint everybody, but there's no ad, sorry.

Steve: You can't get up and pee yet. You've got to keep listening.

Leo: Dammit.

Steve: Okay. So I can't even tell you. Every week there's a most tweeted thing. This time it was me screwing up "The Wrath of Khan" story, of all things. I had said that when Khan was approaching with no contact and radio silence, and Kirk was sitting there with his hand on chin thinking, huh, this is mighty peculiar, I had said that Spock had told Kirk. Well, the first time I saw a comment, it's like, "Oh, of course I knew that." But, you know, one of those.

It was Kirstie Alley playing Saavik, Lt. Saavik. And being junior and very Vulcan, she of course was going by the book. So the fact that Kirk had not raised shields, she was telling him. And then Spock, who's been around the block a few times and on many adventures, he sort of backed Lt. Saavik down, saying "The Captain is well aware of the regulations." So of course after getting blasted by Khan, Kirk said, "You go right on quoting regulations to me." So anyway, thank you, everybody, for the correction. It was the...

Leo: How could you forget Kirstie Alley? My god.

Steve: She was a hot little Vulcan, too. That was - well, anyway. So thank you. And then I loved the second part. Someone else noted that there was another connection between that episode and this podcast, which was that at one point, and it's been so many years since I've seen it, I don't remember the sequence now, but all Federation starships have an override code. And every Starship knows every other Starship's override code. In other words, the ship that Khan was on was using the default password, which was never changed. And as a consequence, the Enterprise was able to log into Khan's ship and bring his shields down and then return the favor of blasting him into space, like he was already there, so not where he got blasted to.

But anyway, I got a kick out of, like, yes. And in all fairness to Khan, he wasn't an Enterprise officer, I mean a Federation officer. He'd been on some dirtball for a long time and was pretty upset about that. So he wasn't sure how his ship worked.

Leo: And, by the way, we have verified with the ICANN wiki that .montblanc did belong to the pen company.

Steve: Ah. Now, that's got to be the definition of a vanity domain.

Leo: It's also a long vanity domain. Seven letters is significant.

Steve: Yeah, exactly. And I have news, Leo. The guy who did the really nice artwork for us where Riker and Jean-Luc's faces were removed and ours were put in their place, he has requested a photo of me with no moustache so that he can update...

Leo: Oh, very important.

Steve: So that it's not looking too dated. So thank you very much.

Leo: Very important.

Steve: Okay. I got a nice note from Steve Holden about SpinRite. Apparently [dropout] Eurythmics fan. And he said: "I wanted to let you know that SpinRite was able to help me recover some 11-year-old [so that's 2006] podcasts from the Eurythmics." And then he gives me a URL to them. So thank you for sharing that, and I think that's cool. And I have a couple of closing-the-loop pieces. Yeah, exactly, there's the link.

Leo: They only had four episodes.

Steve: Ah. They're probably not very long, either.

Leo: No, 17 minutes, 26 minutes, 13 minutes, and 19 minutes. That's not even half of one Security Now!.

Steve: So I've got one tweet saying that "I assume you would recommend LastPass authentication over others," meaning the LastPass app. And I just wanted to note to Mark and everybody else, no. The fact that I'm a fan of LastPass as a password manager does not mean by default, and in this case is not the case, that I like their authenticator. You like Authy. I'm using Google's Auth. Although I've downloaded but haven't yet found time or the need to configure another one that I talked about last week, was probably what prompted Mark's question.

But no, LastPass, the problem I had was that there wasn't enough on the screen. That is, I like to have a bunch of websites on the screen without having to scroll a long way. And LastPass's UI was just taking up way too much space. So I thought, okay, that's, you know. And they're all the same. As long as they're written well and they've been maintained, it just doesn't matter which one you use.

Leo: Yeah. My concern was, and I don't know if the LastPass Authenticator does this, but if it syncs the authentication information back to LastPass, that would obviate the whole point. I have two-factor. One of the points of having two-factor, which is if somebody got my LastPass, it's a single point of failure now. If somebody got my LastPass bundle, they would, you know, they can still use the Authenticator. Now, I don't know, I didn't check, but I just thought, you know, let's get some heterogeneity in the security here and not use two products from the same company.

Steve: That's an important word, yes. Oh, and I was also talking about XON/XOFF. And I was referring to it as, you know, like teletypes, where it was being used - and XOFF was sent by the computer to stop the paper tape reader so the computer could catch up and then start up again. And Simon Pickup said, regarding SN-626, he said: "XON/XOFF live on." He says: "I use it every day pressing Ctrl-S to pause scrolling on the tty output," you know, T-T-Y, teletype output, "of an exterm." He says: "Surely you do, too." And I'm thinking, I'm trying to think if there's any terminal I have that runs so slowly now that that actually is useful. I mean, you know, we use like more pipe or something in order to...

Leo: I think it's the other way around, that it runs so fast you can't keep up, so you're pausing it.

Steve: That's my point is that, like, by the time you enter the command, you're at the bottom, you're at the end of the output. And it's like, okay, fine. So anyway...

Leo: Some keyboards, though, I think, still do have - maybe it's the Scroll Lock or something.

Steve: No, it's just Ctrl-S to Ctrl-Q.

Leo: Oh, Ctrl-S and Ctrl-Q. Yeah, I used that for years. I remember that, yeah.

Steve: Yup. Yup. And so, and they work today, but nothing is so slow that you have any time to move your hand over to the keyboard, or like get the Ctrl key down by the time it's done.

Leo: Yeah, you've got to use more or less or something like that.

Steve: Okay. And Leo, you'll get a kick out of this. I cannot tell you how many people, fewer than corrected me about Lt. Saavik, sent, "What the hell is getting your car smogged?"

Leo: People don't know what that is?

Steve: We have an international audience, and I saw that over and over and over last week.

Leo: Oh, that's interesting, yeah.

Steve: So I didn't - yes. So for those of you who don't have, who aren't in the U.S., here, with CO2 and other nasty chemical emissions from the tailpipe of our cars, in order to renew the annual registration, it's necessary to prove that your car still meets the regulations and the guidance for emissions of smog, thus getting your car "smogged," as it's called in slang, for the year model. So I bought a Jeep back in '84, in the early days of SpinRite, which believe it or not is still on the road. It's a six-cylinder inline, and it refused to die, so I sold it to my best friend for a buck. I said, "Here you go, here's your spare car. I'm not going to sit around and wait for this thing to die."

But the point is that it's not clean, but it's old, so it only has to be held to the emissions requirements at the time, so it's all kind of grandfathered, and that's fair. But anyway, a huge number of people heard me and said, "Getting your car smogged?" And when you think about it, that would make no sense at all unless you...

Leo: It's an "e-test" in Canada. They call it "e-test," apparently.

Steve: Ah.

Leo: And I would imagine that there are something along those lines...

Steve: Equivalents, yeah.

Leo: ...in emissions tests in other countries, as well.

Steve: Maybe not in China. I don't think China has one.

Leo: Can you see the smoke? Okay, well, maybe you should get that fixed.

Steve: Yeah, when you have to wear a mask to walk around outside it's - yeah. In that case you're the one who's getting smogged.

Leo: And there are states that apparently don't do this. Somebody in the chatroom's saying Wisconsin doesn't do it.

Steve: Really.

Leo: Yeah. Here in Cal...

Steve: Oh, that's right. It's a California thing.

Leo: It may be a California - I think some states probably also do it. It may be a Cal - yeah, Minnesota, Michigan, no. Emissions check in U.K. Florida has no emissions test.

Steve: How about Texas? Does Texas do that?

Leo: Texas does not, no.

Steve: That was my guess when it came to mind.

Leo: I'm guessing, yeah. Anything goes there. By the way, you have a right to carry a sword now in Texas, so that's good.

Steve: Well, you know, there might be snakes in the wild.

Leo: You never know.

Steve: That's right. Okay. So in what is probably the most awkward of, like, okay, how can we justify this, I mean, Heartbleed, that was not a big stretch. That kind of worked. MSBlast, that named itself. This one uses the AT&T at the end of Sharkn, and then add "o." So it's SharknAT&To. Okay. We knew they were going with Sharknado, fine. They did win the podcast, the once and only time this will ever be called the Sharknado podcast. Have you seen any of those previews, Leo?

Leo: I have never, I am proud to say, seen one "Sharknado" movie.

Steve: Oh, I couldn't watch the movie. But even the previews, even the commercials are just...

Leo: But they're intentionally bad; right? I mean, they're not trying to be good.

Steve: Yes. Like Leslie Nielsen in "Airplane." I mean, well, you've got sharks flying through the air, biting people. And it's like, okay. Fine. So, but regardless of that, unfortunately a ridiculous name was given to a serious problem. And so to offset the ridiculousness of the name, we had the serious problem.

When I encountered this in doing the research for today's podcast, and the near criminal negligence this demonstrates, or should be regarded as, on the part of AT&T, I was stunned. And it immediately became our focus for this week, and unfortunately the title, not only as yet another useful walkthrough showing just how much can go wrong, but also to advise this podcast's listeners themselves who may be AT&T U-Verse subscribers and might therefore be directly affected; and to continue to support the explicit intention of this publication, who did the research to further highlight AT&T's negligence and the extreme danger the negligence has placed their paying and trusting customers in. So, yes, Sharknado indeed.

The paper is so well written. I corrected a few typos here and there that I found, but as I was reading it I was thinking, wow, this is written in a very, well, this is an English-speaking voice. And as we know, we've covered some in the past that weren't. So I'm just going to share this. I did cut out some and edit it down for length. But it's beautifully put together.

They write, and of course I've got the link in the show notes: "When evidence of the problems described in this report were first noticed, it almost seemed hard to believe. However, for those familiar with" - [dropout] or I'm sorry, "the technical history of Arris" - and that's what you keyed in on, Leo - "and their careless lingering of hardcoded accounts on their products, this report will sadly come as no surprise. For everyone else, prepare to be horrified," they wrote.

"In all fairness, it is uncertain whether these gaping security holes were introduced by Arris, the [dropout], or if these problems were added after delivery to the ISP, in this case AT&T U-Verse. From examining the firmware, it seems apparent that AT&T engineers have the authority and ability to add and customize code running on these devices, which they then provide to the consumer, as they should.

"Some of the problems discussed here affect most" - in fact, there's one that affects all - "affect most AT&T U-Verse modems regardless of the OEM, while others seem to be OEM-specific. So it's not easy to tell who is responsible for this situation. It could be either; or, more likely, it could be both. The hope behind writing this is that the problems will be swiftly patched and that, going forward, peer reviews and/or vulnerability testing on new releases of production firmware will be implemented prior to pushing it to the gateways." Again, so that's a summary of basically let's [dropout] and clean up their act and do this right from now on. So we can hope.

"Security through obscurity is not acceptable," they write, "in today's high-threat landscape. And this is especially true regarding devices which, A, route traffic, sensitive communications, and trade secrets for millions of customers in the U.S.; B, are directly reachable from the Internet at large; and, C, have wireless capability and therefore have an additional method of spreading infection and releasing data. Regardless of why," they write, "when, or even who introduced these vulnerabilities, it is the responsibility of the ISP to ensure that their network and equipment are providing a safe environment for their end-users," meaning, yes, the responsibility unquestionably is AT&T's.

"This, sadly, is not currently the case. The first vulnerability found was caused by pure carelessness, if not intentional all together. Furthermore, it is hard to believe that no one is already exploiting this vulnerability at the detriment of innocents." I mean, they are that bad. Shodan is, like, already knows them all. So this is horrific. "Which is why," they write, "this report is not passing Go, not collecting $200, and is going straight to the public domain. The vulnerabilities found here will be ordered roughly from least to most prevalent." There's four of them.

Okay. So we've got a little bit of trouble with attribution. We're not sure if it's Arris or AT&T. But regardless of where the problems were injected, where this happened, they wind up being in the victim's home.

Okay. The first one, so we're going from least to most, believe it or not, SSH is exposed to the Internet with the superuser account hardcoded with a username and password. They write: "It was found that the latest firmware update" - not an old one, like the one now - for two of these modems, the NVG589 and NVG599 modems enable SSH - which of course we know as the secure shell service and server, meaning that an SSH client anywhere can connect to your U-Verse modem - enable the SSH server and contains hardcoded credentials - okay, get this, and they're here, right here. "Remotessh" is the password. I'm sorry, "remotessh" is the username. The password is "5SaP9I26." There you have it. Anybody want to log into someone's U-Verse modem, scan the 'Net for SSH services and try using "remotessh" as the username and "5SaP9I26" as the password, and maybe you'll get in.

So they write: "...using those credentials, which can be used to gain access to the modem's cshell client over SSH. The cshell is a limited..."

Leo: Oh, that's nice. I'm glad they use "cshell."

Steve: Isn't that convenient, Leo.

Leo: I far prefer that to "bash."

Steve: You don't want to have to grab the text and the reference to figure out, like, how do I blank the line? Yes, yes. Wow. "Cshell," they write, "is a limited menu-driven shell which is capable of viewing/changing the WiFi SSID and password, modifying the network setup, reflashing the firmware from a file served by an FTP server anywhere on the Internet, and even controlling what appears to be a kernel module whose sole purpose appears to be the injection of advertisements into the user's unencrypted web traffic."

So if you do go to a website that's not over HTTPS, as we know, you have not established a secure tunnel that transits through the router. So if this module were in use, it would be able to install U-Verse's own ad banners into pages as they were returned from the remote server. "Although," they write, "no clear evidence was found suggesting that this module is actually being used currently, it is present and vulnerable. Aside from the most dangerous items listed above, the cshell application is also capable of many other privileged actions."

To reiterate the carelessness of this firmware's release - remember, the most recent firmware - the cshell binary is running as root, and so any exploitable command, injection vulnerability, or buffer overflow will result in a root shell. He says: "Yes, it is running as root and trivially susceptible to command injection through the use of the menu's ping functionality; and, due to not sanitizing parameters, one can execute arbitrary commands through the menu." That is, you use the semicolon to append two commands. The ping accepts, you know, ping, and then the IP address you want to ping.

Then you put a semicolon and, like, echo /bin/nsh and pipe that to /etc/shells. It's like you're executing Unix commands as root by adding whatever you want to on the end of the ping command, where it's expecting an IP. You just give it a little bit more. Incredible. So then, after doing that, and they give some examples here, you exit and then reconnect via SSH. The prompt you get changes, and you now have a root shell. You type pound, you know, bang, I'm sorry, not pound, bang, the exclamation point, and you receive a busybox root shell of AT&T U-Verse routers everywhere. Wow.

And then he adds: "Please note that the cshell binary was only examined briefly" - not that he needed to examine it any further because he already had everything you could want - "and only until the easiest exploit was found." So they thought, no, let's just try adding a shell command to the end of the ping's IP. Oh, what do you know, that works. Okay, we're done here. Wow.

So they say: "Judging by the binary's repetitive use of unsafe C functions, one can guess that hundreds of additional vulnerabilities most likely exist. However, we find it highly amusing that the first vulnerability found was so trivial that it looks like it came out of one of those hacking tutorials that were so popular in the '90s." I mean, it's like it's so bad, it's like from an example in a textbook. Don't even ever think about having, like, not checking your parameters to see if it's actually an IP address and only an IP address that's been handed to the ping command. And these people didn't.

Censys, which is an Internet security firm, reports 14,894 IP addresses having hosts which are vulnerable. They say this is no guarantee, there's no guarantee expressed or implied in terms of this number will be all inclusive. Almost 15,000 people discoverable on the Internet currently have this port listening, waiting for someone, an anonymous anybody, to connect. Which will then allow them to take over your residential router with root privilege and do anything they want. And this is now in the public. Everybody knows.

Leo: Yeah, you just gave it out.

Steve: Yes, exactly.

Leo: If you were a bad guy, you knew it already.

Steve: You knew it already. This has already been published. And what we need to do is make sure AT&T doesn't screw around for a few months deciding how they're going to cover their tail and whether or not they want to fix this or not. I mean, pressure needs to be applied. And I agree with their position. This is not a small mistake. This is criminal negligence. I wouldn't be at all surprised if a class-action suit is filed, and it ought to be.

Leo: So it wasn't Arris. It was AT&T, which has the rights to modify Arris firmware for these U-Verse modems.

Steve: Yes. Well, what they said at the start was because both parties are involved - and we don't know. We don't have clear attribution. So it could have been built in, or it could have been added, or [crosstalk].

Leo: But it's not present in other Arris modems? Because I use an Arris modem. I think a lot of people use Arris cable modems.

Steve: That one is present in two. When we get to number four, it's in all.

Leo: Oy.

Steve: So we're going in order of increasing pervasiveness of the problem. Okay. Second problem. That was [dropout]. Second problem. [Dropout] called "caserver," C-A-S-E-R-V-E-R, which is an HTTPS server present in the NVG599. That was one of the two. The previous one was the 589 and the 599. This one is in the 599. They write: "An HTTPS server of unknown purpose was found running on port 49955 with default credentials. The username 'tech'" - T-E-C-H, just in case anyone wants to go out sporting around - "and an empty password field conferred access to this highly vulnerable web server, which used only a basic authentication scheme," that is, nothing fancy. You just have to give it "tech" and an empty password, and you're in.

"The server seems slightly unstable with its authorization capability, denying access on the first attempt even with valid credentials, and eventually completely locking up with an 'unauthorized' message. It remains unclear," they write, "whether this is just poor coding or more security through obscurity, but either is unacceptable. And," they write, "there's a trivially exploitable command injection vulnerability here." So first it's open, we know what the credentials are, and it's exploitable.

They say: "The exact intended purpose of the caserver is unclear, but its implications are not. Caserver is an HTTPS server that runs" - and I was immediately wondering, okay, wait a minute, an HTTPS server, that means it had to have a private key. And we're talking about a private key in a horribly insecure modem. So that's never a good idea, but okay. Maybe keeping its traffic encrypted was more important. But there's, like, other ways to do encryption than over TLS. So laziness abounds here.

"Caserver is an HTTPS server that runs on port 49955 of affected devices, which seem," they write, "to only be the NVG599 modem. Caserver script takes several commands, including upload a firmware image" - so, yes, anybody can replace your firmware if they like.

Leo: That's a good one. I like that. That's a useful - that'd be - come in handy, yeah, yeah, yeah.

Steve: Be handy for field service, Leo. "Requests to a get_data handler which enumerates any object available to its internal SDB databases with a lot of fruitful information," they write, "and requests to a set_data command which allows changes to that database to be made." And they show a screenshot of the request which caused the command injection. And so it's all documented here. Again, as the root user. And again they note that, for the first request the server will probably reply, "You are not authorized to access this page." They say: "This can simply be ignored, and resubmitting the request shown will yield command execution." And, you know, change the firmware.

Leo: This can't be Arris. This has to be some nitwit at AT&T.

Steve: It's incredible. Yeah, you're right, it's too egregious.

Leo: Yeah.

Steve: The service can be a little quirky, they say. It locks up as I said before, after about five requests, blah blah blah. Okay. So I'm going to skip the rest and go to number three: Information Disclosure/Hardcoded Credentials. The next vulnerability involves a service also running, in this case on 61001, which will give an attacker a plethora of useful data about the device. The attacker, however, will need to know the serial number of the device ahead of time. So not so bad. Once this information is acquired, the request can be made.

On the other hand, consider that AT&T U-Verse knows the serial number of every device every customer has. And so if law enforcement or intelligence services had a valid reason, they could get the serial number and, with assistance from AT&T, could use this in order to attack the device. Well, the mixed blessing of this I'll get to in a second.

So they provide the format of the command of what you do on port 61001. You have to provide this weird set of hex characters, 001E46, and then a serial number in order for it to authenticate. When the correct serial number, the OUI is the Arris organizationally unique number, and username and password are submitted, and they document that, the server will hang for several seconds before returning a response. Afterward, several pieces of invaluable information are returned about the modem's configuration, as well as its logs. So it says, here you go. Here's everything you could ever want to know. The most sensitive pieces of information are probably the WiFi credentials and the MAC addresses of all internal devices in your network. And they could be used to exploit the next vulnerability.

Oh, forgot to mention, for anyone who's interested, the hard-coded username and password credentials are bdctest/bdctest. So again, username and password, both are the same, bdctest, lowercase. They write: "This is the second-most vulnerability at the moment. It is not the biggest threat since the modem's serial number is needed to exploit it. This may change if an attacker were to find a reliable way of obtaining the serial number. If present, an attacker could use the aforementioned caserver to retrieve the serial number." Ah, I forgot that, yes. So the previous vulnerability which anyone can exploit does return the serial number, which you're then able to leverage for use in the third vulnerability to get the log dump and a dump of all of the useful information that you might - about what's going on. Just incredible.

Number four, and this one no one is going to believe, the most prevalent of all. They write: "The most prevalent vulnerability, based solely on the high number of affected devices, is the firewall bypass" - that is to say the incoming traffic NAT router bypass. We talk about NAT routers as being inherent firewalls because unsolicited traffic [dropout] to an initial outgoing traffic, so it's not return traffic, is blocked by default. So we all have firewalls. This bypasses that for any external attacker, allowing them to access the devices in your home.

So they write: "The most prevalent vulnerability, based solely on the high number of affected devices, is the firewall bypass that is made possible by the service listening on port 49152." This program takes a three-byte magic value \2a\ce\01 followed by the six-byte MAC address and two-byte port of whichever device your VCR, your webcams, your Ring Doorbell, you name it, whatever they want that's behind there would like to connect to from anywhere on the Internet. What this basically means is the only thing protecting - and remember, this is the most widespread one. Apparently they're all affected. The only thing protecting an AT&T U-Verse internal network device from the Internet is whether or not an attacker knows or is able to brute-force the MAC address of any of its devices.

However, the first three bytes, that is, the first six characters of the 12-character MAC address, as we've often discussed, the MAC address is 48 bits, so it's 24 bits of the manufacturer and then 24 bits of a unique serial number for the MAC address to make them globally unique. They are very predictable since they correspond to the manufacturer, which in this case would be Arris. Maybe AT&T changes it themselves, but probably not. Or maybe it's the manufacturer of the submodem subsystem, if that were the case. Given this, an attacker could start out with this scheme with the unknowns marked.

And so basically they show us a template for that starting up, the \2a\ce\01. And then they use an \ab\23\ed, which is the known high 24 bits of the MAC, and then you have to guess the other ones. On the other hand, there aren't that many. But they say: "To make matters worse, this TCP proxy service will alert the attacker when they have found a correct MAC address by returning a different error code to signify that either the host didn't respond on the specified port or that an RST, a reset packet, a TCP reset was returned. Therefore, the attacker is able to attack the MAC address brute-force and port brute-force problems separately, greatly decreasing the amount of keyspace which must be covered."

So essentially this is an access method by MAC address. It allows a remote attacker to scan for the MAC addresses of the devices behind the router, essentially performing a device scan on your LAN from anywhere on the Internet. And this, I have to say again, is present in all AT&T U-Verse modems.

And they have some more examples of ways to exploit it that they used. And they get to: "At which point it is now feasible for a determined attacker to use a brute-force attack. Aside from the brute-force approach, there are other methods for obtaining the MAC addresses such as the previously mentioned vulnerability." I mean, notice how these things even link together? Like for number three you need data from number two, and now for number four you need data from number three. Like almost like this was a plan, the way these exploits, the data provided links them together. You have both plausible deniability, but it ends up being horribly frightening.

So they said: "Aside from the brute-force approach, there are other methods of obtaining the MAC address, such as the previously mentioned vulnerability, or using a wireless device in monitor mode in order to sniff the wireless client's MAC addresses." Good point. Remember that, if you did promiscuous sniffing within the area of the user's WiFi network, the MAC addresses are not encrypted because they have to be on the outside of the packets in transit for them to get from point A to point B.

So anyone passively obtaining any network's traffic will, after a few minutes or half an hour, have eventually seen all the devices that are periodically transiting, some keep-alive traffic or beacons or pings or whatever, and obtain a list of all - now, that does require you to be local for a while. But as they're saying, you could also use the method from the previous exploit in order to have it dump out the log of all the devices that are talking to it right now.

So they say: "Going off of the example above, if the device's MAC address," and they give it, "has an HTTP server running on port 80," which, you know, some device in your house, "and the attacker wants to connect and issue a GET request on the web root, the command will be," and they provide it, where essentially you are connecting to TCP servers running which are normally protected by the NAT router that lets you cut right through them.

And they say: "This will open," in their example, "an unauthorized TCP connection between the attacker and the 'protected' web server, despite the user never authorizing it." They say: "It is believed that the original purpose of this service" - that is, essentially, the ability to break through the de facto firewall and get to a device behind it, and their presumption makes total sense - "that the original purpose of the service was to allow AT&T to connect to the AT&T-issued DVR devices which reside on the internal LAN." That would make sense because they would know what the MAC address is, and that would allow them to immediately poke right through and access a device at a known MAC address at a known location.

"However," they write, "it should be painfully obvious by now that there is something terribly wrong with this implementation. Added to the severity is the fact that every single AT&T service observed has had this port" - I've got to rewind. "Added to the severity is the fact that every single AT&T device observed has had this port 49152 open and has responded to probes in the same way. It is also important to note that the gateway itself cannot be connected to in this manner." So only devices behind it in the home, on the LAN. "For example, an attacker cannot set the MAC address to that of the modem's LAN interface and the port to correspond to the web configuration console. This attempt will fail. This TCP proxy service will only connect attackers to client devices behind."

So they wrap it up, saying: "In conclusion, in 2017, when artificial intelligence runs the largest advertising firm on the Internet; when only last year the largest leaks in American history occurred; and where vehicles are self-driving, autonomous, Internet-connected, and hacked, why do we still find CGI injections, blank default passwords with root-privileged services exposed and what will most likely be termed 'backdoored' credentials?"

He writes: "Developing software is no trivial task. It is part of this company's core services. But carelessness of this magnitude should come with some accountability. Below are some workarounds for the vulnerabilities described in this write-up. The time of full disclosure is gone, mostly; but let the time of accountability begin. Accountability. Or is it okay to continuously accept free credit monitoring from vendors and governments and corporations who have 'accidentally' exposed your privacy; and, in this case, maybe that of your family's, too?"

So a nicely put together piece and documenting just a horrific situation. I would hope that our listeners, this podcast's listeners have their own NAT router behind their AT&T U-Verse box. That solves most of these problems. If you don't rely on its NAT routing, but rather have the Arris device, just plug it into one of our little favorite $49 EdgeRouter X's, and then you're done. You're safe. There's still the problem of somebody getting into the Arris modem itself. That they could do.

But if anyone is concerned, and I didn't keep going with this because I just wanted to cover this, but the workarounds are there. And the good news is the outcry that this has to produce will force this to get fixed soon. But in the short term, I would immediately make sure you have your own NAT inboard of the AT&T U-Verse "router," unquote, and that way at least your internal network is protected. Somebody could, using that earlier, but also much less widespread - the big problem is the widespread one. So it was like, as they said, every single device they looked at had that port, 49152, open, listening, and with that proxy service running. So you want your own NAT inside. And as we know these days it's not difficult to do that.

Leo: Well, but if you're using U-Verse TV service, you may not be able to NAT it.

Steve: Oh.

Leo: That may be a bigger problem. I'm not sure. I seem to remember that people who use U-Verse have to use the router provided by AT&T, the TV service.

Steve: Ah. In that case, maybe there's a way to run your own LAN, put your TV on...

Leo: Put your LAN separate from your TV, yeah.

Steve: Yes. Yes, yes, yes, yes.

Leo: And it sounds like, it's not clear, I mean, from reading the Nomotion post, they do ding Arris for previously having problems like hardcoded passwords in the firmware. But it's not clear if this affects Arris modems on other than U-Verse systems. I'm still not quite getting whether that's the case or not.

Steve: I completely agree. We don't know. But AT&T U-Verse is large footprint.

Leo: Yeah. Oh, absolutely. It's, yeah, number two or number three. But, yeah, I would guess that most of the people who listen to this show have their own router in between them and the modem.

Steve: If not, then now.

Leo: I would hope so, yeah.

Steve: There's never been a better lesson for why it's always good to do that anyway.

Leo: Yeah. Well, again, I once again apologize for the sound quality in this show. I think what we're going to start doing, Steve, from now on is a double-ender. I'll give you a week to figure out how you can record your audio at your end and then send it to us. And that would solve the problem, not for the live folks, but it would solve the problem for the podcast.

Steve: Good. I will do that. And I should mention I've been watching. I've still got the traceroute running. And it's actually not three hops down. That was sort of an initial artifact. It looks more like, I don't know if there's, like, if there's a problem everywhere. But it looks like it's the first hop. It's like the other end of my cable connection is not returning. So I'll just run through all my stuff again. And I'm able to do this. During one of our sponsorship breaks, I wrote down the IP that is the other end of this Skype connection, and I'm going to start poking, I'm going to probe to it. And anyway, so it'll be fixed next week, one way or the other. I'll get it nailed.

Leo: Okay. But, yeah, worst-case scenario we'll do a double-ender, and that way at least the podcast will sound okay. And I do heartily apologize to everybody. And for those of you missing one word in five, they'll all be in the transcript. Well, maybe not.

Steve: Well, if Elaine can figure it out...

Leo: The only one that was weird, and I just want to make sure Elaine gets it right, is when you said AngelFire, it sounded like the "g" was missing. And that changes the meaning entirely. So I just want to make sure that Elaine understands it's AngelFire with a "g."

Steve: Oh, yes. I had to wait for a minute to figure out what it was you meant, but now I understand.

Leo: Yeah, think about it, you'll know.

Steve: Now I understand. You really don't want fire there.

Leo: No. Steve is at, and the podcast, too, GRC.com. It's where you'll find the world's finest hard drive maintenance and recovery utility, of course SpinRite. If you have a hard drive, you need SpinRite. Everything else freely available including transcripts of the show and audio, 64Kb audio. We do audio and video at our website, TWiT.tv/sn. You can get that on demand.

Watch live, though, if you're around Tuesday at 1:30 Pacific, 4:30 Eastern, 20:30 UTC. A lot of people watch the show live, like to get the latest security news hot off the presses. And if you do do that, join us in the chatroom. Always a good bunch of people talking behind the scenes at irc.twit.tv. Thank you, Steve. We'll be back next Tuesday. It'll be my last time, and then I'm going on a couple weeks' vacation. But I'm sure that you'll have a good time with Father Robert.

Steve: We'll do it, and I'll talk to you next week, friend. Bye.