Research: Useful & Otherwise

Description: This week we discuss another terrific NIST initiative, RSA crypto in a quantum computing world, Cisco's specious malware detection claims, the meaning of post-audit OpenVPN bug findings, worrisome bugs revealed in Intel's recent Skylake and Kaby Lake processors, the commercialization of a malware technique, WannaCry keeps resurfacing, Linksys responds to the CIA's Vault 7 CherryBomb firmware, another government reacts to encryption, the NSA's amazing GitHub repository, more news about HP printer auto-updating, a piece of errata, some miscellany, and some closing-the-loop feedback from our listeners.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-618.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-618-lq.mp3

SHOW TEASE: Time for Security Now!. Steve Gibson's here. Lots to talk about, including some bugs in OpenVPN. Nothing to worry about. It's been patched. And Steve may question a little bit the methodology. It's all coming up next, plus our image of the week, which was the same image of the week a couple of years ago on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 618, recorded Tuesday, June 27th, 2017: Research: Useful and Otherwise.

It's time for Security Now!, the show where we cover the latest in security and privacy and how stuff works with Steve Gibson. He's the majordomo at GRC.com. Hi, Steverino. Live long and prosper.

Steve Gibson: Hey, Leo. Yes, great to be with you again, Episode 816. What am I saying, 816 - 618.

Leo: Yeah, don't rush it because you've already said you're retiring at 1,000.

Steve: A little dyslexia there, 618. I titled this one "Research: Useful and Otherwise" because there were a couple really interesting, significant stories about some really interesting research and then one that's like, okay, what, you know. And actually, so I think it's going to be a great episode. We're going to talk about another terrific NIST initiative. They're the ones who ran contests that selected the Rijndael block cipher as

the AES standard. And then later they did the same thing for the cryptographic hash function, choosing Keccak for the SHA-3 cipher. Anyway, they're doing another one for a very interesting category that I'm glad for.

Many people ask about quantum computing and what that's going to do to crypto. So our friend Dan Bernstein, who's a well-known cryptographer - in fact, it was his elliptic curve that was the original inspiration for the SQRL project that I embarked on. He's examined the nature of RSA crypto, which we know is based on the multiplication of primes and the difficulty of then factoring them back out of the product of the primes in a quantum computing world. Unfortunately, we have Cisco's, what I've described as "specious malware detection claims," which is the "otherwise" in this research. It's like, oh, okay. I've dug into it, and I don't know what's behind it, but we'll cover it.

And then the meaning of post-audit OpenVPN bug findings. Also some worrisome bugs that have just been revealed in Intel's recent Skylake and Kaby Lake processors, what they mean and what our listeners can do. The unfortunate commercialization of a malware technique. The fact that WannaCry keeps resurfacing. Linksys responding to the CIA's Vault 7 CherryBomb firmware that we recently discussed. Another government reacting to encryption. The NSA's amazing GitHub repository. More news about HP printer auto-updating. A bit of errata, some miscellany, and then, of course some closing-the-loop feedback from our listeners. So I think another great and interesting podcast here.

Leo: Indeed, indeed, as always. We've got a live studio audience. Bruno's visiting from Switzerland, James and Robert from Missouri. So they're in the studio, too.

Steve: That's much better than a dead studio audience.

Leo: Yeah.

Steve: More animated.

Leo: We've had the dead studio audiences. It's no fun at all. All right. I see a Picture of the Week. In fact, I saw it earlier on Twitter.

Steve: Yeah, so did everybody. And the problem is - I put it up because I wanted to acknowledge everybody who sent it to me. But when I looked at it, I thought, eh, this looks very familiar. Now, I figured, had we shown it before, I would have read it to the people who are listening to the audio. So I went to GRC.com/sn.

Leo: Oh, you are so smart.

Steve: And I put in, "In this corner we have," and bang. Two years ago in June, so in 2015, this was our Picture of the Week.

Leo: And if you look at the date on it, I think it's from 2006. Can that be?

Steve: Yes, it does have an '06 copyright on it, too.

Leo: Wow.

Steve: And this is one of the things that we see on the Internet from time to time is these things sort of come back into vogue. Someone rediscovers it and posts it, and it goes viral and so forth.

Leo: Well, but also, 11 years later, it's still true.

Steve: It is, interestingly, still relevant. So for those who don't see the picture and who may not have been listening for two years, it's a boxing ring. And in one corner we have someone holding up a big data security sign with lots of equipment there to create all this secure infrastructure. And the announcer holding the microphone says, "In this corner we have firewalls, encryption, antivirus software, et cetera. And in this corner, we have Dave."

Leo: And Dave's wearing a sweatshirt that says "Human Error."

Steve: Yeah. And the point being, yes, you can bring all of this technology to bear, and Dave can still defeat all of your best intentions just by being himself.

Leo: Another way you can tell it's dated, you see antivirus software. We've been having a lot of discussions on the radio show. People have been begging me to ask you, and I'm sure we've covered it on the show, your position on using antivirus software.

Steve: I've listened to it all over the weekend. And you're 100% correct in everything that you said. It is the case - and in fact, Tavis's most recent shower was it brought about his findings that even Defender has problems that have been fixed. The one thing you didn't say, and it's arguably too technical for your weekend audience, is that in order for third-party AV to do what it must, it's necessary to hook deeply into the operating system. And that fundamentally destabilizes it. They do the best job they can, but they're coming along, and they're having to put their hooks in, literally, to the kernel.

And so the advantage that Microsoft's built-in system has, in addition to just being there and built-in and sanctioned and so forth, is it's been - the OS has been tested with it in place. Microsoft isn't testing their OS with other third-party AVs in place. The AVs are having to test theirs against Microsoft. So it just seems all around better and more stable not to put a third-party AV platform in. Which is exactly what you were saying.

Leo: On Windows 8 and 10 you've got Defender. It's not the best in the world. Nobody says it is. But the problem with antivirus software is it gives you a sense of false confidence because no antivirus software can protect you from everything. In fact, most protect you from a fraction of the total threat.

Steve: Yes. And I would argue that it's getting worse because, in this spy vs. spy cat-and-mouse game, the strategies to avoid detection are evolving, and the AV people are always responding. They're not preemptive. They're unable to detect something that doesn't exist yet. So they're inherently lagging, and the malware people are just trying to keep a few steps ahead. So unfortunately there isn't - as we have said, the only solution, I mean, this is the classic, do you blacklist or do you whitelist? The only solution is to whitelist, that is, to only allow what you know to be good to run.

Leo: And that's Windows 10 S, which I think is dubious security.

Steve: Well, which has just been defeated badly, yes.

Leo: Right. But the problem is they don't whitelist aggressively enough. They allow Win32 subsystem. They allow Office.

Steve: Well, yeah, I heard you talking about Win32 subsystem over the weekend. It has to be there for legacies. There's too much stuff that still runs it.

Leo: See, that's what I'm thinking is I think the only thing that they really have it in there for is the centennial apps, which are wrapped Windows 32 apps that they can then put in the Windows Store. They're not true UWP apps. And the biggest one is the one they have to have, which is Office. So you've got 32-bit Office, and so that's the vector. And I just - anyway.

Steve: It'll be interesting to see how it goes.

Leo: Yeah. I just feel like you give up so much with Windows 10 S because you can't run a decent browser, and it's of dubious value in terms of security.

Steve: Yeah, well, and you will be giving up SQRL's ability to prevent...

Leo: Oh, really, yeah, because SQRL's not in the store unless they...

Steve: Yes. Well, SQRL's not in the store. But also one of the questionable things they've done is they've prevented the browser from having access to the localhost IP. And SQRL uses that in order to perform its client-provided session antispoofing protection. And Windows 10 S is the only system on the planet that doesn't allow that to happen. It still works, but you just lose that extra...

Leo: But you do understand why they did that. I mean, again, that's what you give up for the security.

Steve: Yeah.

Leo: All right.

Steve: So the NIST, a branch of that, the Information Technology Laboratory, the ITL, announced an initiative that they've been working toward for a while. In this initiative's introduction they said: "The world is increasingly interconnected by myriads of devices." Should "myriads" be plural?

Leo: "Myriad" is one of them singular plural...

Steve: I thought so, "...by myriad devices working in concert to accomplish important tasks including," they write, "automotive systems, sensor networks, healthcare, distributed control systems, and the Internet of Things, also cyber-physical systems and the smart grid." They write: "But these devices may have resource limitations compared to common desktop computers. For example, they may have significantly reduced power consumption, less computation power, and orders of magnitude less memory than desktop computers. These constraints can make it difficult to implement modern cryptographic algorithms, most of which are designed for desktop or server environments.

"To address this issue, different cryptographic algorithms have been tailored for resource-constrained devices. The academic community has performed a significant amount of work on this type of cryptography, called 'lightweight cryptography.' This work includes efficient implementations of conventional cryptography standards and the design and analysis of new lightweight algorithms and protocols.

"In 2013, NIST's Information Technology Laboratory started a lightweight cryptography project to investigate the issues and then develop a strategy for the standardization of lightweight cryptographic algorithms. In 2015 and 2016, NIST held two lightweight cryptography workshops to solicit feedback on the constraints and limitations of the target devices, the requirements and characteristics of real-world applications of lightweight cryptography."

And now, recently, NIST decided to create, through an open process, a portfolio of lightweight algorithms. ITL, which is this group within NIST, has published NIST Internal Report 8114. And I have a link to the full report in the show notes for anyone who's interested, which is titled the "Report on Lightweight Cryptography," to "summarize the findings of this project and to outline NIST's plans for the standardization of lightweight algorithms. Here, we present highlights from this report, especially the devices targeted by lightweight cryptography, how the algorithms were designed, and the standardization of lightweight cryptographic algorithms." And of course I'm not going to go into all that detail.

But the beginning of their actual report says, which is sort of interesting because it helps to give us a better sense for exactly what this is, they say: "This report provides an

overview of lightweight cryptography, summarizes the findings of NIST's lightweight cryptography project, and outlines NIST's plans for the standardization of lightweight algorithms. In particular, NIST has decided to create a portfolio of lightweight algorithms through an open process.

"This report includes a list of questions to the stakeholders of lightweight cryptography that will serve" - we need to somehow shorten that. It's a little bit of a tongue twister. Lightcrypt or something, who knows - "that will serve as the basis for determining requirements. NIST will develop profiles based on community responses to these questions. These profiles are intended to capture cryptographic algorithm requirements imposed by devices and applications where lightweight cryptography is needed. Algorithms will be recommended for use only in the context of profiles, which describe physical, performance, and security characteristics."

So anyway, they then talk about the target devices. They enumerate and clarify the performance metrics which are their goals, talk about the cryptographic primitives - and we've discussed them all on this show in the past - block ciphers, hash functions, message authentication codes, and stream ciphers. And the idea being they want to take all the lessons that we have from traditional full-strength heavyweight crypto and say, okay, if you've got a light bulb, and you want crypto in a light bulb, or especially if you have some little battery-powered tag somewhere, and you want cryptographic strength, what are the constraints in these applications, thanks to the nature of the device, and also remember the cost. Cost is a factor, and larger chips cost more.

And so there's some pushback against the need to be able to accommodate the existing cryptographic primitives which just assume they're in a multi-gigabyte desktop environment or a server. So, for example, things like reducing the block size of a block cipher. We've talked about this before, how AES is deliberately 128-bit block because what the symmetric cipher does is map every single combination of 128 bits into a different and unpredictable combination of 128 bits, one for one, so that for every input combination of 128 bits, you get a different output 128 bits. And that's reversible, so you can go the other way. And what's so cool, I mean, almost magical, is that that mapping function is controlled by an input key. And what AES gives us is officially three different key sizes which can be used, which allows you to have variable strength of key within a given block size.

And so it's important, I mean, one of the things that they clearly recognize is there are tradeoffs for making reductions for the classically established algorithms and standards. But there are also things we can do to offset the tradeoff. For example, in a block cipher, most block ciphers have this notion of rounds. We've talked about this, and we did a whole podcast on AES, for example, where we actually looked in detail at an AES round, the idea being that each cycle mixes the bits in itself in a reversible way.

But if you just did that once, there just isn't enough mixture, enough mixing that a single round creates, making it easy to reverse the process or to figure out what the key must be. So to overcome that, they do multiple rounds in order to - so each round is reversible, so the whole thing ends up being reversible. But by the time you're done with, like, 14 rounds, you're not only exhausted, but you're so scrambled up, even though they're reversibly scrambled up, there's just no way to get back to the beginning.

So, but the point is, the design was made deliberately with the tradeoffs of a desktop environment in mind. So, for example, in AES there are these things - throughout crypto there's a thing called an S-box with is essentially, it's a little fixed mapping box. It takes some number of bits in, often eight, and it produces eight bits out. So that's like a fixed key kind of mini cipher. And so those S-boxes are scattered around. But each S-box,

because it's taking eight bits in and needs to map those to a different eight bits out, that's an expensive process in terms of processing time or hardware real estate, if you wanted to integrate this on a chip.

So, for example, if you went to a four-bit S-box, where it only needs to map four bits into a different four bits, that's incredibly simpler in terms of die-size. The four-bit S-box is estimated to have a relative size of 28, just sort of a numeric integer, 28; whereas an eight-bit S-box is 395. So the problem is the requirement scales exponentially with the bit length of the S-box. So the point is there's a whole different problem than has been really addressed before in the notion of bringing high security to low-resource or resource-constrained environments. And, for example, where you might make each individual round in a cipher less strong, you can end up achieving the same end strength by just doing more rounds. And since light bulbs and IoT devices tend not to be performance constrained, the tradeoffs can be different. You can say, okay, we've got more time to do this on our IoT device. So we're going to, for example, crank the rounds up really high while reducing what each round does, which creates huge savings in chip real estate, RAM requirements, and so forth.

So anyway, I think this is a great initiative. I mean, these guys, it was exactly this kind of open process which we need somebody to be the honcho of, somebody to manage and spearhead - as I said, that's where we got the Rijndael selection as the AES cipher, and we got the SHA-3 next-generation ciphers. I think this is great. And all for the best. But I would note that having a suite of carefully considered lightweight cryptographic primitive building blocks does not automatically make IoT devices secure. That is, having them isn't going to change things overnight.

What this sort of falls into the classification of necessary - so it's necessary, but not sufficient. Having approved cryptographic standards which are appropriate and still secure, just the tradeoffs were made differently, removes one of the largest objections that IoT makers can present. They could say, well, we'd like to have crypto, but it's going to make our stuff uncompetitive because it's going to raise the price. And besides, we all know that consumers say they want security, but nobody's actually willing to pay for it. Well, if we have a suite of lower, probably lower performance, but parity secure and appropriate primitives, then cost stops being an objection. And although it's not going to fix today's devices, I love the fact that we'll have this in place for tomorrow's. And then nobody can say that it's going to make these things cost more because it won't. It'll be crypto for free, essentially, in the same devices. So, nice piece of forthcoming research.

Another piece of cool research is this whole question of - the term is "post-quantum RSA," meaning, okay, we had sort of this uncomfortable sense that quantum computers are on the horizon. They're getting more capable. They're inching forward. It's a huge topic of research. The question is, how will the presence of quantum computing impact present-day encryption? So a bunch of academics, Dan Bernstein being among them, I think there's three others, decided to really tackle this question.

So in the abstract of their paper, and I've got the link to the full PDF in the show notes for anyone who's interested, they say: "This paper proposes RSA parameters for which key generation, encryption, decryption, signing, and verification are feasible on today's computers, while all known attacks are infeasible, even assuming highly scalable quantum computers." Okay. So this paper proposes RSA parameters for doing everything RSA needs to do. And I'll look at this question of feasibility in a minute because what they had to do, okay, feasible, maybe. But the point being that they've looked at the question of can we keep RSA, is it possible to keep it, even in the presence of highly capable quantum computers.

So they said: "As part of the performance analysis, this paper introduces a new algorithm to generate a batch of primes. As part of the attack analysis, this paper introduces a new quantum factorization algorithm that is often much faster than Shor's algorithm" - Peter Shor is a mathematician who famously, about 23 years ago, back in '94, he described how future quantum computers, as we've assumed they were going to operate, could be put to the task of prime factorization. And this is one of the things that upset everybody is it looks like, oh, crap, quantum computers are going to be much better factoring machines than existing sequential machines that we have now. So Shor's algorithm is often cited as this classic example of, whoops, quantum computers are going to kill crypto. But in the process, Dan and his group came up with something even better - often, as they write, much faster than Shor's algorithm and much faster than pre-quantum factorization algorithms.

Okay. So as we know, it's been the longstanding intractability of the integer factorization problem, that is, the difficulty of determining the two prime number factors which comprise a public key. And that is the entire basis for RSA asymmetric key technology. In RSA, as we've discussed in the past, the public key is the product of two secret prime factors. And the reason a public key can be made public, even though it's got the secrets in it, it's composed of two prime numbers that have been multiplied. That's all that the public key is. So the reason the public key can be made public is that, even knowing it, we do not currently know how to break it back apart into the two prime factors that were multiplied in the first place to obtain it.

So with factorization, this is a place where size matters. We know, for example, the prime factors of 35. That's not difficult. Any grammar school kid can tell you the prime factors of the number 35. But it's easy to do that only because there are not many possible prime factors smaller than 35, and not many possible solutions to it. This is not true when the products of primes are several thousand bits long. We know that primes are surprisingly common throughout the number space. They do not get less common sort of counterintuitively since a prime by definition is a number not divisible by anything smaller than it. You kind of think that, as they're getting bigger, there are just, like, so many numbers smaller than it, that it would be increasingly difficult to find them. Turns out that's not the case. There's lots of them forever.

So as a consequence, as the primes get really big, there are just too many possibilities. And no one has found, to date, a highly efficient way on non-quantum computers to break apart a sufficiently lengthy product of two large primes. Thus RSA stands, although we have been inching the public key size up. We were at 1024 for a while. Now we're at 2048. And eventually we'll be at 4096, just because everybody likes powers of two. And in fact it's the relative weakness of the factorization problem which accounts for the fact that, for example, an RSA public key needs to be 2048 bits long, whereas an elliptic curve public key, which uses a different hard problem, not prime factorization, it can be 256 bits long and with equal strength.

So the essential question this paper seeks to answer is, even assuming the presence of fast quantum factoring, and also while offering this new faster quantum factorization algorithm that they've called GEECM, under quantum factorization how does the difficulty of usage, that is, using a given RSA public key, how does the difficulty of using it scale relative to the difficulty of a practical attack using even quantum factorization as the prime factors' product length increases.

Okay. So we know, for example, a quantum computer could probably cut through today's 2048-bit RSA public key without breaking a sweat. So they wanted to understand how RSA scales. That is, if we need to make factoring much more difficult because quantum computers can factor much better than current-day computers, and for some reason if

we want to hold onto RSA, how does it scale? So they ask the question, is it actually true that quantum computers will kill RSA?

And they write: "The question here is not whether quantum computers will be built or will be affordable for attackers. This paper assumes that astonishingly scalable quantum computers will be built, making a" - the term they use is "qubit," that is the term - "making a qubit operation, a quantum-bit operation..."

Leo: Q-U-B-I-T, not...

Steve: Right.

Leo: ...the biblical measurement.

Steve: Right, right, as in a quantum-bit operation as inexpensive as today's bit operation. So they say: "Under this assumption, Shor's algorithm easily breaks RSA as used on the Internet today." Yup, dead. "The question is whether RSA parameters" - meaning prime lengths - "can be adjusted so that all known quantum attack algorithms are infeasible while encryption and decryption remain feasible." So that's what I meant when I said, you know, how does it scale? Can we keep growing the public key size so that we could still use it, yet very fast quantum computers can't crack it.

So they write: "The conventional wisdom is that Shor's algorithm factors an RSA public key 'n' almost as quickly as the legitimate RSA user can decrypt." Okay, so basically it really weakens the benefit. The point is that using the key is about the same level of difficulty or speed as cracking the key. So they say: "Decryption uses an exponentiation modulo n; Shor's algorithm uses a quantum exponentiation modulo n. There are some small overheads," they write, "in Shor's algorithm, but these overheads create only a very small gap between the cost of decryption and the cost of factorization.

"The main finding of this paper is that standard techniques for speeding up RSA, when pushed to their extremes, do create a much larger gap between the legitimate user's costs and the attacker's costs." Meaning the cost to attack increases much faster than the cost to use. So RSA could be kept alive, kind of on life support, if it was like, if we had no better solutions. Now, it's worth noting we've already got post-quantum crypto. We haven't talked about it yet on the podcast because qubits, there aren't a lot of qubits wandering around yet. None of us have them on our desks or in our pockets or anything, but they're coming. And so academia is ahead of this. So this is not, you know, no one should misunderstand this as suggesting that it's practical, or we actually will be using RSA in a post-quantum era. It's just sort of interesting research. And I think useful research, too.

So they say: "These extremes," that is, pushing RSA to its extreme. "These extremes require a careful analysis of quantum algorithms for integer factorization. As part of this security analysis, this paper introduces" - and then they talk about their own new faster quantum factoring solution. And then finally they say: "We report initial implementation results" - so they actually, you know, the rubber meets the road here. They built this stuff under assumptions about what would happen if, they said, okay, if quantum factorization ends up being essentially on par with current, what does it mean? So they said, and I had to do a double-take on this because it's like, wait, what?

"And finally," they say, "we report initial implementation results for RSA parameters large enough" - that is, remember, long keys - "large enough to push all known quantum attacks above 2^100 qubit operations." Right? So like we're going to say that qubits are efficient, so we're going to force you to use a whole ton of them, 2^100. So this is very much like the numbers we're used to seeing: 2^128, 2^256, okay, this is 2^100 qubit operations, making them again incredibly time consuming and thus impractical. So they say: "We report initial implementation results for RSA parameters large enough to push all known quantum attacks above 2^100 qubit operations." And here it comes. "These results include successful completion of the most expensive operation in post-quantum RSA, namely" - Leo, are you well centered over your ball?

Leo: I am.

Steve: "...the generation of an eight-terabit public key."

Leo: I like it. Eight trillion bits. So the big ones now are 4,096 bits.

Steve: Yeah, yeah. We don't - we're using 2048, you know, 2,048, and that's just fine now.

Leo: I think eight terabits, even quantum computing might be hard pressed.

Steve: Yeah. So as I said, this is not meant to - we're not suggesting that this is practical. But they're saying, okay, we just kind of wanted to go there. We wanted to say, okay, we got fast quantum factoring. How badly does this kill RSA? And the point is, okay, if for some reason you absolutely positively must use prime factorization as the hard - you still have to use it as the hard problem to solve. And nobody is suggesting that. But if you did, well, an eight trillion-bit public key, that would do the job. Of course, you know, schlepping that around, communicating - oh, and of course one bit goes bad, I mean, it's difficult in today's storage environment even to reliably store an eight trillion-bit public key. It itself would have to have its own error correction, which would increase its size substantially, just to protect against internal bits changing within the key, just sort of spontaneously, for quantum reasons.

So, yeah, at this point, that's a problem. There are all kinds of post-quantum crypto which is ready to go. Lattice techniques are looking like they're becoming very popular, but there's a handful of them. So I just thought this was fun, to say, okay, can we keep this thing alive? Well, yeah. Is anyone going to? Uh, no. We're going to be switching to different crypto when quantum machines happen.

Now, that was all kind of fun and useful research. I was really interested initially to see how this claim, which was made last week in a public presentation by Cisco, could be substantiated. They claimed, Cisco did, that they would soon be commercializing and making available to their enterprise customers, and I will end with a cautionary note here in a minute, the ability to detect malware within encrypted web traffic with 99% accuracy. And I thought, okay, what? So I did some digging.

Last October at the AISec 2016 Proceedings, which was the 2016 ACM Workshop on Artificial Intelligence and Security, two Cisco guys, Blake Anderson and David McGrew,

delivered their research paper titled "Identifying Encrypted Malware Traffic with Contextual Flow Data." Okay, now, first of all, that sounds okay. Interesting, but it's not clear that that's the same as within encrypted web traffic. So one of my complaints is that somebody at Cisco has decided they want to sexify this research and have everyone go, ooh. But, okay, contextual flow data is not malware within encrypted web traffic. It's behavioral, and it's metadata.

Anyway, so the abstract of this paper from last October said: "Identifying threats contained within encrypted network traffic poses a unique set of challenges." Yeah, amen, baby. "It is important" - they didn't say that, I did. "It is important to monitor this traffic for threats and malware" - again, no disagreement there - "but do so," they write, "in a way that maintains the integrity of the encryption." Okay. So we've of course often talked about the middleboxes, as now the term is, which is an enterprise-class proxy that itself is the encryption termination point. So it's got a certificate, and it is synthesizing remote site certs on the fly and signing those. And the reason everyone in the enterprise trusts the certs that it has just made and signed is that all the devices in the enterprise have had its public key stuck into their certificate root store specifically so that no alarms will get raised, and everybody trusts those certs.

So the point is Cisco is saying, okay, but we want to do this without decrypting the traffic, which immediately says, "Huh? What? Okay, you've got my attention." They said: "Because pattern matching cannot operate on encrypted data..." Right. As we all know. Well, good encrypted data is indistinguishable from pseudorandom noise. They say: "...previous approaches have leveraged observable metadata gathered from the flow, e.g., the flow's packet lengths and inter-arrival times."

And remember we've previously encountered and reported on this podcast numerous side-channel attacks on encryption which leveraged content-dependent compression ratios. That is, if the plaintext was compressed, then encrypted - which is the sequence you need because you can't compress encrypted data. It won't compress. So the problem was that the ratio of encryption is inherently a function of what data you're - I'm sorry. Yeah, the ratio of compression is inherently dependent upon what data you're compressing. And so encryption doesn't change the size. Thus you can infer the compression ratio from the encrypted size. And then, if you're really clever, you can, through lots of repeated test encryptions, you can reverse-engineer some unknown plaintext when you pad it with known plaintext. That we've all covered in the past.

But they're saying, okay, we're not going to do that. "In this work," they write, "we extend the current state of the art by considering" - and then they have invented a term called "data omnia" approach, which I guess means taking lots of inputs. "To this end, we develop supervised machine learning models that take advantage of a unique and diverse set of network flow data features." Okay, so, yes, data omnia, as they term it. "These data features include TLS handshake metadata, DNS contextual flows linked to the encrypted flow, and the HTTP headers" - meaning nonencrypted headers - "of HTTP contextual flows from the same IP source address within a five-minute window. We begin by exhibiting," and blah blah blah.

Okay. So basically what they've said, what the marketing claim is absolutely bears no resemblance to what they're actually doing. I was tempted to, like, dig into the paper further. But I thought, you know, it was behind a paywall, didn't seem like it was worth $15 to go any further.

They do, however, say: "We begin by exhibiting" - this is still in the abstract, so this is what's publicly available. "We begin by exhibiting the differences between malicious and benign traffic's use of TLS, DNS, and HTTP on millions of unique flows. This study is used

to design the feature sets that have the most discriminatory power. Then we show that incorporating this contextual information into a supervised learning system significantly increases performance at a 0.00% false discovery rate for the problem of classifying encrypted malicious flows. We further validate our false positive rate on an independent real-world dataset."

Okay. So they have a 0.00% false positive discovery rate. And the first that came into my mind yesterday as I was putting this together was, okay, but so does a can of dog food. You can set any can of dog food down in the middle of a desk. You can even plug it in, if you want to. It doesn't matter either way. It will also have a 0.00% false positive discovery rate. Malware can be zipping past, and no matter what happens, it will never generate a false positive malware discovery. What their paper is silent on, is my point, is nowhere do they talk about the false negative detection rate, which is to say, how much malware whizzes past which they miss.

So I did find - and I already had, actually - a full paper that the same authors produced earlier last summer. So that allowed me to go into it. That paper was titled "Deciphering Malware's Use of TLS Without Decryption." And this paper, it looks like what they're now claiming is a result of the fact that they couldn't actually do what they were claiming to do. And the paper that is fully public shows them trying to use TLS handshake metadata to infer the content of the subsequent connection. And we know about TLS handshake metadata. That's a client and a server negotiating cryptographic protocols - which security suites are available, which TLS extensions do they support and so forth.

So they did a lot of research. I mean, and I'm not knocking them for doing research. We need research. And we need research to be published so that people can look at it and learn from it. The problem I have is when the marketing people get a hold of it and decide that telling the truth isn't going to sell anything except dog food. And so they end up saying something, they make a claim that is patently impossible and which no one can deliver. So the original research showed that, if they took, I don't know, I didn't count, but it was like 10 or 15, somewhere like that, different families of malware, and they examined the TLS handshake metadata which those families of malware used, given a universe of known malware, they could discriminate with 90% accuracy which among that malware the handshake metadata was.

Okay, so that's an entirely different problem than finding malware in a sea of benign traffic. So this is useful. But I noted that they weren't using DNS metadata originally. Now they are. And of course using DNS, looking at the DNS queries that a network makes, will go a long way toward telling you if something is malicious because it's, you know, malicious domains are themselves, by their name, suspicious looking. We see what command-and-control servers use, you know, 64 characters of gibberish.ru, for example. Okay, that would raise some suspicion. And DNS is not encrypted, as we know, so it's always available for inspection.

So anyway, the problem is that Cisco is claiming that they're going to be deploying this soon. Cisco's Senior VP David Goeckeler said at a press event in San Francisco a week ago, last Tuesday, "We get both privacy and security," adding that the new technology, he says, can detect malware with 99% accuracy. Okay. Maybe they have a system that can do that, but it is not by looking inside of non-decrypted TLS traffic. It is by looking at everything else. And the fact is TLS, the actual metadata, we know from their previous work is providing almost no useful information. I would imagine, just from the abstract in the paper which closely precedes Cisco's announcement that they have some massive breakthrough, is that they're probably looking at DNS largely, using that as most of the source of input to the system.

So anyway, my advice, if anyone is a Cisco customer with an enterprise profile, this has every earmark of being a low success rate heuristic which is attempting to do something certainly desirable and sexy, but basically impossible. So don't rely on it exclusively at first. Test it thoroughly in a network lab setting before signing any long-term contract. I would say, you know, approach this one with skepticism because it just - it'd be nice to do. Maybe by looking at everything else like DNS you could develop some useful heuristics. And you don't want them false-positiving too much because that's going to, of course, create a problem. And, boy, I would be a fan of not needing to decrypt, as everyone knows, our HTTPS traffic at the enterprise border. It's just a cleaner solution all around not to have to do that. So if they can pull it off, bravo. But I would say approach with caution.

So the unfortunate title of this work is "The OpenVPN Post-Audit Bug Bonanza."

Leo: Uh-oh. Oh, dear. Everybody uses OpenVPN. Everybody. Right?

Steve: Yes. And I believe should. Guido Vranken has a full report in his blog on WordPress. I've got the link to it in the show notes. But to set the context of this, I need to share his claim. So he writes in the summary: "I've discovered four important security vulnerabilities in OpenVPN. Interestingly, these were not found by the two recently completed audits of OpenVPN code. Below you'll find mostly technical information about the vulnerabilities and about how I found them, but also some commentary on why commissioning code audits isn't always the best way to find vulnerabilities." And I'll certainly be responding to that, as well.

He says: "After a hardening of the OpenVPN code as commissioned by the Dutch intelligence service AIVD and two recent audits, I thought it was now time for some real action," he says. "Most of these issues were found through fuzzing." He writes: "I hate to admit it, but my chops in the arcane art of reviewing code manually, acquired through grueling practice, are dwarfed by the fuzzer in one fell swoop. The mortal's mind," he says, "can only retain and comprehend so much information at a time; and, for programs that perform long cycles of complex, deeply nested operations, it is simply not feasible to expect a human to perform an encompassing and reliable verification."

He writes: "End users and companies who want to invest in validating the security of an application written in an [and he has in quotes] 'unsafe' language like C, such as those who crowdfunded the OpenVPN audit, should not request," he says, "a manual source code audit, but rather task the experts with the goal of ensuring intended operation and finding vulnerabilities, using the strategy that provides the optimal yield for a given funding window."

Now, just know I disagree with pretty much everything he's said so far. But there's value in this. He finishes: "Upon first thought, you'd assume" - oh, no, this is me now. So okay. So he says, you know, fuzzing rocks; code audits fail. So upon first thought you'd assume both endeavors boil down to the same thing. Oh, no, I'm sorry, this is him still talking, his voice. "Upon first thought you'd assume both endeavors boil down to the same thing, but my fuzzing-based strategy," he writes, "is evidently more effective." I will argue that in a minute. "What's more, once a set of fuzzers has been written, these can be integrated into a continuous integration environment for permanent protection henceforth; whereas a code review only provides a 'snapshot' security assessment of a particular software version." No argument there. He finishes: "Manual reviews may still be part of the effort, but only where automation (fuzzing) is not adequate."

Okay. So first of all, just to make sure everybody understands, we all know what a code audit is. Fuzzing, as we have discussed in the past, is the art and practice of just throwing nonsense at the API or the communications channel of something and seeing if it breaks. I remember way back, years ago, eEye, the eEye security firm, E-E-Y-E, had a whole lab of Windows machines, and they were finding problems. And so, like, suddenly one of the machines would just stop because their code had thrown some data at a function which it wasn't written to expect, and the system would crash. And so as the fuzzer was doing this fuzzing, it was logging everything it was about to do so that they could rewind and repeat the problem and then go take a look manually at what the fuzzer had discovered.

Okay. So my position is I strongly disagree that either approach is more or less valuable than the other. The two approaches are different, and it's that difference that makes both important. We know from so much coverage in the past on this podcast that there are definitely many classes of critical problems that fuzzing will not find, for example, subtle flaws in cryptographic implementations, secrets-based timing, and power changes that would enable side-channel attacks, unsafe assumptions about the use of fundamental cryptographic primitives, and I could go on like that all day, talking about what it is that a cryptographer sees when they look at this kind of code, that isn't about does it crash, it's about is it worth using, I mean, if when it's working is it secure. So this notion that deliberate code auditing is not every bit as important as fuzzing is utter nonsense.

That said, as we've talked about, fuzzing is a very powerful tool for discovering an entirely different class of very important bugs that could, as Guido correctly asserts, easily slip past code auditors. That's not the kind of bug they're looking for. It's not the kind of bug they really can look for, which is why fuzzing makes absolute sense in conjunction with code auditing. And as he points out, where he talks about a "given funding window," he's right. Fuzzing is super cheap because it's automatable. And it's super easy to do once it's been set up.

So, yes, I'm delighted that he did this. I'm delighted that as a consequence OpenVPN is even more solid than it was after the code audit, which made it better. And now we've got the best of both approaches. But no one should think for a second that any money could have been saved by only using fuzzing instead of a careful code audit by cryptography-savvy coders and developers who know how to take a look inside of this and make sure that the things they can see were done correctly. There are things they can't see, and that's where fuzzing can be very effective. So hats off to Guido for doing this. And I understand that he's proud of his results, and he should be. But it's not the case that one replaces the other.

Leo: Should we be concerned about OpenVPN? I mean, did he find bugs that are, in fact [crosstalk]?

Steve: Yeah. Oh, yeah, yeah, yeah. I'm glad you asked, yes. It has been fixed.

Leo: Good, okay.

Steve: So, I mean, there were several crashes and something that may have been leverageable to remote code execution, so the typical kinds of things that are problems. His fuzzer crashed OpenVPN, and he found four different problems where you could double-release some memory. And although he didn't take it to an exploit, he said,

"Oops," you know, "that's a bug in the code. Let's fix that bug." So the updated version of OpenVPN is already available. And I'm sure everybody who has their repositories being checked for updates, I looked and I saw that the FreeBSD server at GRC has an OpenVPN update waiting for it. So, yeah. So it's important to update OpenVPN now because we just found some more problems. But I didn't want anyone to come away thinking, oh, look, that audit was BS, and we should just turn everything loose to fuzzing. It's like, no. We need both.

Boy, and speaking of fuzzing, it turns out, and this is a problem, Intel's Skylake and Kaby Lake processors have been found to contain serious microcode bugs. We know that low-level processor flaws are not without precedent. But fortunately they are extremely rare; and our chip vendors generally, I would say, do an amazing job dealing with the stunning complexity of modern Complex Instruction Set, CISC, C-I-S-C, processors. So far Intel has been relatively quiet about the problem, but this is in the news. Reporters who have talked to engineers at Dell and Intel have been told that the problem exists, and there are fixes for it that are just waiting for confirmation before being pushed. There is a microcode patch which is currently being quietly tested to make sure that it doesn't break anything else.

And naturally, being a processor-level bug, it is OS agnostic. All operating systems and other software running, any software running under the OSes - and that would include VMs, potentially, so Windows, macOS, Linux, FreeBSD, et cetera - can be vulnerable. A quietly published Intel chip errata reads: "Under complex micro-architectural conditions" - meaning the micro architecture that is what the microcode drives which implements the higher level instruction set, which is what the code sees and runs - they write, "...short loops of less than 64 instructions that use the AH, BH, CH, or DH registers, as well as" - and by the way, that's the high eight bits, the AH, BH, CH, DH, the high half, the high eight bits of the A, B, C, and D registers - "as well as their corresponding wider register." RAX which is the 64-bit AX, EAX which is the 32-bit view, or AX or AH may cause, they write, "unpredictable system behavior." Yeah, as in the instructions don't work right.

They say: "This can only happen when both logical processors on the same physical processor are active." In other words, this is a hyperthreading problem. As we know, Intel, before they went to multicore, they said, hey, you know, if we take one physical processor and allow it essentially to have two program counters, that is, so that we could be doing two things at once, we could get - we could, like, squeak out marginally more performance. It wasn't, like, double. And, frankly, it wasn't very much. I don't remember now what the numbers were, like 10% more. I mean, you know, it was free. So it was like, okay, not bad, as long as it's free. And that's what the so-called hyperthreading is, where essentially they create two virtual processors from one by allowing two things to happen at once. Well, it looks like Skylake and Kaby Lake, but not, for example, Haswell, the predecessor family, have a microcode flaw in the implementation of hyperthreading which affects short loop repetitions of the registers of some instructions.

So, okay. So there is a Debian page which has the best discussion of this. I have the link in the show notes. In fact, it's so important and so good that I made the first show-numbered bit.ly link in a long time: bit.ly/sn618. So that just bounces your browser, redirects your browser to this Debian.org page, where they've got beautiful coverage of this. What happened was that some developers at the beginning of the year, back in January, were working on a compiler for the OCaml language, which is an obscure kind of crazy-looking - I took a look at it because, like, what's OCaml? And I still don't know. But it's amazing. And as a side effect of the way it was driving the downstream GCC compiler, it set some constraints on GCC that were unusual, that caused it to produce problems.

And so these guys are thinking, oh, crap, our OCaml compiler isn't working. And they looked at the code, and they said, uh, yes, it is. The chip must be broken. Which of course they couldn't believe because that just doesn't happen. Turns out they were right. And so their compiler was inducing this flaw in Skylake and Kaby Lake chips, which - and I'm not sure of the sequence of this. They went back and they discovered that even as far back as the end of the first quarter of last year, this could be occurring on Skylake chips. So Intel knows about it. There is firmware kind of like somewhere. And I imagine that what we'll end up seeing is a firmware patch that Apple will push out with macOS, Microsoft will push out to their users of the affected processors, and the various Linux and Unix and so forth, everybody's going to get this, end up getting this patched. The good news is firmware is patchable.

Leo: How does microcode get patched, though? Because that's hardware.

Steve: No, it actually is patchable. You're able to patch the microcode in Intel's chips. It's been done before. There was a firmware...

Leo: It doesn't modify the chip. It must, like, somehow interrupt the call or something; right? Or no.

Steve: I've not looked into it. They're calling it a firmware update.

Leo: Interesting, huh.

Steve: So I don't know. It wouldn't surprise me if they load something from a store at power-up and then allow it to be patched. So anyway, I imagine sooner or later we're all going to be getting this. The Debian page suggests - and I don't know that anyone has to take action on it. First of all, if you don't - if you have Haswell or earlier, you're not affected.

Leo: That's what they do, they fix the microcode. That's why you have microcode. It interposes between the actual physical hardware, and it lives in memory.

Steve: Yeah.

Leo: It lives in special high-speed memory. Well, that's interesting.

Steve: Yeah, yeah. And we did, for anyone who's interested, we did a whole series years ago on modern processor architecture and how it all works. And the incentive for microcode, because microcode began in the...

Leo: [Crosstalk]; right?

Steve: Well, it began in the minicomputer era. For example, the ARM chips don't need microcode because they are RISC processors, that is, they have such a simple instruction set that you can implement just in hardware. But Intel took the earlier path. They have a complex instruction set. And so it's so complicated that you can't implement it reasonably in hardware. So you have to create a computer within a computer. Basically the microcode is interpreting the complex instructions to get the work done. And so this is a benefit of, or kind of a mixed blessing, but in this case it is fixable. And so you can patch it in order to change its behavior at a very low level.

Leo: Early on in the 8086 they had - remember the floating point era.

Steve: Yup.

Leo: It did floating point math wrong.

Steve: Oh, and you were able, I remember, you were able to bring up a Lotus or an Excel spreadsheet.

Leo: You could demo it easily, yeah.

Steve: Yeah, and put in a couple special numbers, and [buzzer sound], it gave you the wrong answer.

Leo: Yeah.

Steve: It's like, ooh, that's not good.

Leo: Yeah.

Steve: Yeah. Okay. So I said at the top of the show that a malware technique had been commercialized. And this is unfortunate. Gizmodo picked up on this, and I did a little bit of deeper research into a company called NaviStone, N-A-V-I-S-T-O-N-E. And they are, you know, they're everything I object to in terms of the way a company should operate. They are just the king of slime. Okay. So you visit a website and begin filling in a contact form, or maybe the site's Create an Account form. Or maybe start filling in purchase information. Or maybe your own browser, which has been paying attention in the past, beats you to the punch on some fields and autofills a bunch of them for you, being your little handy-dandy assistant.

In the traditional world, where we all grew up, this was a passive process, meaning that your browser was showing you a page. There was a form there. And somewhere on the page, generally at the bottom below the fill-in area, there would be a Create an Account button, or a Submit button. And the implicit and longstanding rule has been nothing happens with any of that information until and unless you press the button to explicitly and deliberately send that provided information off to the hosting web server of the

website you're visiting.

And, Leo, you'll remember a few months back we talked about it, in fact you played with during the podcast, a malicious code on browser pages that was deliberately creating hidden unseen offscreen form fields which the browser's autofill automation would fill, even though you couldn't even see it doing it. And of course it was doing it to help you, with well-meaning intent, but that malicious code on that page could suck those browser-filled unseen field contents off the page and spirit them away to parts unknown.

Well, on the heels of the adage "If it can be done, someone will do it," we have "If someone can make money doing it, they will." And in this case we have this what seems to me to be an ultra slimy company named NaviStone. Their public website says, you know, it claims - get a load of the - this is just amazing: "Reach your anonymous website visitors with retargeting postcards and convert 50X more visitors than digital display ads." They have also a slogan: "Convert your anonymous website traffic. Harness the power of consumer intent data. Take your retargeting offline."

Then they say: "Until today, your ability to retarget anonymous website shoppers was limited to low-impact, low-response digital display ads. With the NaviStone turnkey postcard program you can reach out directly to anonymous website shoppers, tailor your marketing to the individual shopper based on their website behavior, mail personalized postcards within 24-48 hours after a site visit." And then in their little "Who We Are" box - because I was just, I was thinking, oh, wow - they say: "Traditional direct marketing contact strategies are driven entirely by past purchase behavior. As that data ages, it becomes less predictive of future responsiveness. At NaviStone we lead the vanguard in progressive website visitor tracking technology." Yes, you can find it here, folks.

Leo: I might a little bit disagree with you. And the only reason I say this is it's very common - this is what it's mostly used for. And you might have even done this. You go to a site, and you load the shopping cart up, but then you don't execute. You leave for whatever reason. And in fact a lot of people do this on purpose because what you'll get later is an email saying, hey, we noticed you loaded the shopping cart, but you didn't buy it. Maybe you'd like to get 10% off or 20% off on these items. And that's primarily what this is used for.

Steve: So you're saying, if you are completely anonymous, and they don't know who you...

Leo: Well, you're anonymous because you didn't submit. So if I go to a website, let's say I go to Zappos. By the way, it's very widely used. If I got to Zappos, and I shop, and I load the shopping cart, and then for whatever reason I leave, maybe I just - I'm not done, I'm going to go back to the shopping cart, I haven't given them any information except the fact that I want to do this. So the next time I go back to that site it can say, oh, I see you didn't submit. Would you like a discount to encourage you to submit? That's the kind of usage.

Steve: Ah. And I don't have a problem with that at that one site.

Leo: Well, that's, see, I don't think it's completely slimy. I think that that's the intent

of this.

Steve: Well, the - okay.

Leo: The assumption you're making which everybody makes is, well, if I fill in a form on the website, the website has no access to it unless I post it, unless I hit the Submit button and do a POST.

Steve: Right.

Leo: So what this is, is some JavaScript that watches the form being filled out and captures it without a post.

Steve: Correct.

Leo: I don't know if that's completely slimy. It certainly could be used in a slimy fashion.

Steve: Well, it sends it to a third party.

Leo: Right.

Steve: So it's not going to the site you're visiting. So the site you're visiting has contracted with NaviStone, and that...

Leo: Very many, by the way, very many sites do this. This is very common in ecommerce.

Steve: Okay. And what they're saying is that you will then receive a postcard or an email, even though you have submitted nothing.

Leo: Well, that's a little dubious because how are they going to get your email?

Steve: That's my point, is this is serious web tracking. This is not just you went to this site, and then you come back, and they say, oh, look, we notice you didn't finish before. Gizmodo tested it. Gizmodo says three sites - Rockler.com, gift site CollectionsEtc.com, and clothing site Boston Proper, they said - "sent us emails about items we'd left in our shopping carts using the email addresses we'd typed into the site, but never submitted."

Leo: That's how they got your email. It was part of what was typed in, yeah.

Steve: Well, yes. Although NaviStone also has your physical address. And what they're selling on their site...

Leo: And how do they have your physical address?

Steve: That's my point. They're tracking, Leo. This is actually real.

Leo: No. They have your physical address if you've entered it into the form. That's where that comes from.

Steve: No. They're a cross-site tracking system. That's what they do. They aggregate this information from across the web, following you around...

Leo: Oh, yeah, yeah, yeah. You may not have entered - okay, that makes sense. Maybe you entered it on Zappos, but you got a postcard from Boston Proper because you'd entered it at one point. But then how do they identify you? They must have some sort of...

Steve: Well, of course. If their JavaScript is also on all these sites, then your browser is carrying their cookie. And so they're able to link you across all these sites.

Leo: So that's why you turn off third-party cookies.

Steve: Right.

Leo: Yeah.

Steve: So anyway...

Leo: I think the problem is that ecommerce cart abandonment rate is about 70%. So what I think completely legitimately companies like Boston Proper are just trying to do is, well, maybe that person wanted this stuff, but for whatever reason got distracted, or the browser crashed. We'd like to remind them.

Steve: And I would have no problem if they kept it in the family. I would have no problem if their JavaScript watched that and then saw you come back. Instead...

Leo: Oh, man. Very few ecommerce sites are self-hosted. You know, they use Shopify and other systems.

Steve: Right, right.

Leo: I mean, that's pretty common.

Steve: And in fact there is a search service, sort of like or reminiscent of Shodan, that's called BuiltWith. And we've talked about it before. BuiltWith is a spider that goes around and looks at, like, it basically harvests the technology that sites use. Gizmodo was a little curious, so they found more than a hundred sites using this NaviStone technology which aggregates that stuff.

Leo: Yeah, yeah. I bet it's many more than that. If not NaviStone, it's somebody else.

Steve: It's a company very much like it, yeah.

Leo: NaviStone agreed to stop doing it; right?

Steve: They backed off as a result of some of this coverage of one aspect of this. But, I mean, it is their business model is to do this.

Leo: Yeah.

Steve: And WannaCry is still making - is bringing tears to some people. There were two instances that occurred in the last week. The Honda Sayama auto plant in Japan stopped their production line of engines and two classes of Hondas, a van and something else, I don't remember, for two days, for nearly 48 hours last week when WannaCry somehow came back and found a couple of the machines that were critical to the assembly line's operation. And then, so that was one bit.

And the other was people were sort of celebrating that Australia's stoplight ticketing cameras were taken down. A network of 55 cameras in Australia got hit by WannaCry, apparently when a technician connected an infected USB drive to one of the devices, and then it quickly spread through the network, which was apparently running Windows OS. So in all of these cases, the moral for us is, if you want your stoplight cameras or your nuclear submarines on a consumer operating system, well, you get what you pay for. So, yikes.

Linksys has responded to the CherryBlossom CIA Vault 7 attack. I'm glad for that. They're a router that many of us have and use. So I just wanted to give a heads-up that last week they, in fact they addressed it directly in an advisory dated 6/21, they said: "Linksys is aware of the CherryBlossom project that was recently released by WikiLeaks' Vault 7 publication. Based on the WikiLeaks report, customized firmware - and of course

we covered this at length last week - was created for certain older Linksys routers without our knowledge or consent for the purposes of monitoring, controlling, and manipulating Internet traffic of a targeted user."

And so customized firmware, they wrote, can be loaded onto a router, either with physical access to the router, proximate access to the router via WiFi, or intercepting the device in transit to be delivered to a user. Which we know apparently has been done in the past. They said: "If users believe their router firmware may have been compromised, Linksys" - and of course how would anyone really know that - "Linksys recommends that users download the latest available firmware" - and then it's just www.linksys.com/support - and update your router.

So, oh, and one thing I noted that I don't think I'd ever seen before. I don't know if this is a recent occurrence. But the advisory finishes, after listing all of the affected model numbers: "We would also like to recommend the following changes after the factory reset is complete to further secure the router: First, set a strong admin password, one that includes capital letters, numbers, special characters, and a password length of at least" - okay, well, they're saying eight characters; but, as we know, more is better here. "Disable guest access if it is not in use. And disable router features like WPS and UPnP," they say, "if they are not being used."

So yay to Linksys suggesting that disabling WPS we know we've long recommended, and Universal Plug and Play, which is of course a real problem in this world where people are getting IoT devices infected, and they can use UPnP to get access to the external world.

Last episode, which was titled "When Governments React," we looked at France, Britain, Japan, Germany, and Russia. And now we have a sixth country. Australia has just said that they intend to push for encryption backdoors at next week's Five Eyes meeting. The Five Eyes alliance, as we know, is Australia, Canada, New Zealand, the U.K., and the U.S., which are a group bound by multilateral U.K./USA agreement for joint cooperation in signals intelligence, military intelligence, and human intelligence. And it's always important for us to remember that any non-technical policy person using the term "encryption backdoors," you know, we have no idea what they mean, well, because neither do they, really. So until we see actual legislation, all we can infer from this is that they're not happy.

In this case, the Australian Attorney General George Brandis is stating he'll be pushing for backdoors, whatever that means, at the upcoming meeting of the Five Eyes in Ottawa, Canada next week, where they will discuss tactics to combat terrorism and protect borders. Australia has made it clear it wants tech companies to do more than to give intelligence and law enforcement agencies access to encrypted communications. Brandis said, in a joint statement: "I will raise the need to address ongoing challenges posed by terrorists and criminals using encryption. These discussions will focus on the need to cooperate with service providers to ensure reasonable assistance is provided to law enforcement and security agencies."

So he has a reputation in Australia. He's previously rationalized away potential objections to backdooring encryption, saying that it is people's tendency to overshare on social media, and that that indicates they don't care if the government or several governments have access to their private messages. So, and there's some grain of truth to that, as I've been arguing lately. We've seen lots of studies that suggest a person will give you their password in trade for an ice cream cone, and that everyone says they want encryption; but when it comes down to it, in order to be truly secure, people tend to be a little lazy, so not going all the way.

I had no idea that the NSA had a repository on GitHub. And it's easy to find: nationalsecurityagency.github.io. And, wow, there are a bunch of neat things. Just to give you just a sense for it, something called Apache Accumulo, a sorted, distributed key/value store that provides robust, scalable data storage and retrieval. It adds cell-based access control and a server-side programming mechanism that can modify key/value pairs at various points in the data management process.

Something called CASA identifies unexpected and prohibited Certificate Authority certificates on Windows systems. That sounds very useful. Control Flow Integrity Research; DCP, a program that reduces the time span needed for making a forensic copy of hard drives for forensic analysis; and on and on. So if anyone's interested, there's just a bunch of really interesting projects: nationalsecurityagency.github.io. So somebody tweeted that to me, and I took a look, and I was really impressed. There's lots of stuff there.

I did get a lot of feedback from our listeners, who followed up on the question of HP printer firmware. Many people tweeted photos, emailed photos, and just sent me notes. So far not, out of maybe a population of 30 or 40 all over the place, not a single HP printer appears to default to firmware update automatic. I think for whatever reason HP must have made the decision that, yes, our printers have firmware. But when they leave the shipping dock, they work. So you could update them if you want to; but they work, so leave them alone.

Unfortunately, as we discussed at length last week, yes, they work; but they've also got very significantly worrisome exploitable firmware because they've got a full Linux subset that can be taken over through various means. So as I said last week, I think it's time everyone should take a moment to update their HP printer firmware, since HP is not doing it automatically.

Oh, and I said something wrong. Ben, who tweeted as @Gingiraffe, said two things: "First, Apple two-factor authentication doesn't use iMessage." And "Apple two-factor authentication," he says, "does have some limited time-based one-time password integration available. Love the show. Cheers." And the moment I read "Apple two-factor authentication doesn't use iMessage," it's like, duh, of course.

Now, which is to say the UI is not iMessage. And what I should have said last week, which is where I completely agree I misspoke, is I said Apple had the advantage of not using the cellular network's existing text messaging platform to send their one-time password tokens because they've got their own system. I said iMessage. That was wrong. I don't know that it doesn't leverage that communications flow. It certainly doesn't have to. And the presentation is different. I don't know, because it is a six-digit code typically, at least in one of the types of prompts and challenges you can receive, maybe it is time-based, where they have a shared key with your device. Who knows. Anyway, thank you, Ben, for letting me correct that.

Leo: Although as we pointed out, if you only have one Apple device, it will use text messaging.

Steve: Because it has no choice.

Leo: It has no choice.

Steve: Right. I got a nice note, or an interesting follow-up to a discussion last week from Grant Taylor, who tweets as @DrScriptt with two T's. He said, and this sort of gave me a bit of opportunity to clarify something, he said: "@SGgrc. Running SpinRite before a format might speed up a format on an otherwise not completely happy drive."

Remember the question I answered was should you run SpinRite before or after. And I kind of said, eh, I really don't think it matters. So I would say that it's true except that the explosion in storage capacity we have witnessed over the last 10 years has changed the entire nature of - and I'm putting it in air quotes here - "formatting."

It was once the case that formatting a drive meant scanning and verifying the drive's storage surface, looking for known and already marked defective sectors. Back then, drives were checked by the manufacturer for "known defects," as they were called, and those were specified and could be entered into the low-level format process to start those sectors off as flagged bad. And then, additionally, the formatting software would just go out, and it would look at all the sectors, and also just kind of make sure they're all present and accounted for. And then, after doing that, it would establish the file system on the drive with those defective areas never put into use, that is, pre-established in the file system as defective. So sectors would fall into clusters, and the free cluster map for the file system would already take those out of service, so nobody could ever use them.

But as drives have gotten bigger, it has become really infeasible to do that any longer. As SpinRite's own struggle to run in a practical length of time has revealed, drives are becoming large relative to any system's, and even the drive's, own ability to actually transfer all of the drive's phenomenal amount of [audio dropout] into the system. And as I've discussed before, I'll be making a huge leap forward in SpinRite's performance with its next release through a combination of pure assembly language code, which will work directly with the hardware, no more BIOS, and leverage the hardware's maximum theoretical data buffer size of 32MB. So this goes from 32K buffers, which we have now, to 32MB buffers, which is the most that the hardware can handle. So SpinRite from v6.1 on will transfer data at the maximum rate, limited only by the drive's rotation; or, in the case of solid-state media, the raw speed of the interface.

But today's drive sizes mean that simple formatting, like what we do when we set up a new drive, is no longer practically able to check the drive's storage integrity. Instead it must assume everything is okay. In Windows we see the so-called "quick format," which makes that assumption. It simply builds the file system's directory structure at the front of the drive and hopes for the best. It skips any actual testing of the drive for the sake of practicality and expediency. And so I would argue that running SpinRite sometime, whether before or after, is probably a good thing to do because otherwise it may be the case that some areas of your drive are never visited. I mean, they weren't visited during formatting. And unless you absolutely fill up your drive, they may stay largely unused for some length of time and not actually be available for use, I mean, not be good any longer by the time you finally get to it.

So again, nobody actually formats a drive any longer. You just can't afford to. They've gotten too big. So you just say, okay, the drive says it's this big. We're going to assume it's telling the truth and that there's all that unexplored territory out there.

And finally, I have a couple of closing-the-loop bits, closing-the-loop feedback from our customers. Jeffry Erickson tweeted that "ProtonMail now has a free and paid VPN service." And he said, you know, "Topic for SN?" And, okay. So VPN services, I think, are more or less generic and, as we know, are widely available. That being the case, I think that certainly more choices and wider feature sets are better. But rapid adoption of a new service just because it's new is probably not the way to obtain the greatest security

guarantee. A VPN service is one place where I think letting things settle down a bit makes a little more sense, all other things being equal. If a new service has some specific feature that makes it especially beneficial for a specific user, like where they are, their terms, whether you're already a subscriber and they're making the VPN service available as part of your existing subscription or your relationship with them, okay. But just standard security wisdom suggests that, you know, give it six months.

Again, there are so many solutions which are time-hardened that I don't know if there's a reason to jump on a brand new one except to notice that it's there, celebrate it, and then wait for the Wikipedia comparison of VPN features to add that service and get populated so that you can easily take a look at it and see if makes sense. So I think it's a good thing. I just don't know that jumping on something brand new, when it's something you're depending upon, makes sense unless you have a specific reason to do so.

Oh, and Leo, I heard you talking over the weekend, answering the question about Steve's favored imaging software.

Leo: Yeah, and I think I got it wrong. I said Drive Image XML. That wasn't it?

Steve: Yeah.

Leo: I just googled it.

Steve: Unfortunately, well, and you did say it's got this annoying generic name.

Leo: Yeah, like Drive Imager or something.

Steve: Like it definitely does. Or like Image for Windows.

Leo: There it is.

Steve: Which is just a little - but you...

Leo: I did buy it, and I use it.

Steve: And it's what I use. I use it on everything. It's from a company called TeraByte, T-E-R-A-B-Y-T-E. It's very affordable. It's $39 or something. And they've got versions for Linux and DOS. They've got packages. It supports UEFI. And what I like about it is that it also knows about NTFS. So you can do single large images rather than the earlier systems that only knew about FAT32 and so they had to break them up into 4GB pieces, which once upon a time that was big. Not anymore. So Dennis, thanks for asking the question, and it's TeraByte Unlimited is the whole name of the company, Image for Windows, and also for Linux and DOS. And they've got a bunch of neat utilities and add-ons and freebies, and they're just a great company.

Leo: Windows 10 does come with its own imaging. I think...

Steve: Yeah, I heard you mention that. Do earlier versions not? Is that a new - is that a Win10 feature?

Leo: I don't know. I'm going to have to ask some people. It might be - that's a good question. I don't know.

Steve: Because I've never been aware - I've always been aware of, of course, backup has always been part of Windows. But that's file level. And as you mention, what we want is something that can boot from a completely dead or an empty drive and then immediately, or given time, reinstantiate the file system, the OS, the other partitions, and everything. And the cool thing about TeraByte's is, for example, you can click on the root of the tree of the drive, which is the drive, the partition table, the various weird partitions that Microsoft now creates. You just say, yeah, give them all to me. And it puts them all in one image. And if you then give it a virgin drive and say, here, put that back, it'll just recreate. Oh, and it's able even to dynamically resize the various partitions to fit the drive that it's going to. So, I mean, yeah, really, those guys solved the problem.

Leo: Yeah. I think it actually started in Windows 7. In fact, it's the older style of backup.

Steve: Interesting.

Leo: And one of the nice things about the Windows version is that all you have to do is put an install disk in, which you've made, of course, because if you don't - you should do that before you do a drive image, do a recovery disk. And then you can run it from the recovery disk. So what you do is you, if you go to Windows key, type "backup," and then it says at the bottom, "looking for an older backup." And this is the thing that I had to kind of figure out. Go to Backup and Restore (Windows 7). And then this has an imaging system built in. So I think this is the Windows 7 imaging.

Steve: Oh, neat. I will check that out.

Leo: It allows you to go to a hard drive, DVDs, a network location. And then you restore it by running the system installer, and you can just restore from the image directly.

Steve: Very nice.

Leo: Which is about everything you'd want, I think, in a Windows [crosstalk].

Steve: I think yes. And I would argue that, for anyone from Win7 on, that makes sense.

Leo: Yea, yeah.

Steve: Nice.

Leo: It's a little hard to find it. It's not obvious that it exists. Somebody had to tell me.

Steve: And now you've told all of our listeners, so that's good.

Leo: Yeah.

Steve: Our frequent contributor, Simon Zerafa, sort of sent a reminder that I wanted to pass on. He says: "Time to check for apps with access to important accounts. Here's the list for Google. Do the same with Twitter and Facebook." And I did. And sure enough, I found some old things that I'd forgotten I'd given permission to access Google.

So it's myaccount.google.com/permissions. Again, just https://myaccount.google.com/permissions. And just look, if you are a Google user, take a perusal over the list. I took about half of them out. I had, like, 15 things. And it's like, oh, yeah, what's that? I haven't used that forever. And it's just good, just makes sense to clean that kind of thing up because no apps typically remove themselves. I don't even know if they can, once given permission. So there's also, of course, Twitter has that and Facebook has that. So thank you, Simon, for the reminder.

And I love this. This is actually the perfect conclusion for this week. This is our "say it isn't true" horrifying observation of the week. Someone tweeting as Klingonveckan said: "Regarding SN-617," which of course was last week, "about eight characters being the most common password length. I just have to bring up the fact, he says, that 'password' is eight characters."

Leo: Not a coincidence.

Steve: It's like, oh, say it isn't true.

Leo: Yeah, not a coincidence.

Steve: Yes, indeed.

Leo: I think the chatroom might have observed that at the time, as well.

Steve: Ah.

Leo: In fact, because of your comments last week, I made some new passwords this week, and I made sure they were all odd-numbered lengths. And when the site said you can have up to 30, I would make it 29, right, because 30 would then be more common than 29.

Steve: Yup.

Leo: So the idea is to kind of foil the statistics, I guess.

Steve: Very good. Don't follow the pack.

Leo: Yeah. Be your own person. Well, if you listen to this show, you probably are your own person, and you probably drive your friends and neighbors crazy. But that's because you know more than they do, thanks to this guy right here, Steve Gibson. Go to GRC.com to find Steve's stuff. It's all there, GRC.com, including SpinRite, the world's best hard drive maintenance and recovery utility, his bread and butter. Information about SQRL is there, Perfect Paper Passwords, Password Haystacks. ShieldsUP!, lest we forget. How many millions of people have used that now? I mean, it goes up at an amazing rate. I always, when I set up a router, that's the first place I go. GRC.com. He also has the show there, audio versions of the show, plus Elaine's transcripts. There'll be one or two places in this show where she doesn't know what you said. Just weird, just little dropouts. But it wasn't bad. It was better, for sure.

Steve: Good, I'll do it from now on.

Leo: Yeah. And - sorry, Elaine. If you want audio or video, you can also get it from us, TWiT.tv/sn. And if you have a podcast app that you favor, you can always search for Security Now! in that and subscribe, and you'll get each and every episode. And that's always nice. We will be back next Tuesday, 1:30, right after MacBreak Weekly, 1:30 Pacific, 4:30 Eastern, 20:30 UTC, if you want to tune in live. Join us in the chatroom, irc.twit.tv. You hear them referenced quite a bit.

Steve: I think actually in this case we will be back in two weeks, my friend.

Leo: Oh, you know, you sent me an email, and I wanted to talk to you about that. Yeah, next week is the Fourth of July. And TWiT is dark. But, I mean, if you want - I know you hate missing shows. And I know your audience hates it when you miss shows. So if you wanted to record the day after or the day before, something, we could do that. Otherwise, enjoy a little break.

Steve: Well, I had a back-and-forth with Lisa, and she said, "No, we're taking the day off." So she said, "Enjoy..."

Leo: Yeah, we are. You can't record on Tuesday. Nobody will be here. But if you wanted to, I could set it up for another day.

Steve: Let's take a day.

Leo: Think of it as the all-star break. Just relax.

Steve: We'll enjoy the Fourth of July, and we'll resume with lots of news in two weeks.

Leo: Maybe nothing will happen.

Steve: That's right. That's right.

Leo: That'll happen. Thank you, Steve Gibson, and we'll see you next time on Security Now!.

Steve: Okay, my friend.