
# NSA Fingerprints

**Description:** This week we discuss a new clever and disheartening abuse of our browsers' handy-dandy username and password autofill, some recent and frantic scurrying around by many OS kernel developers, a just-released MacOS zero-day allowing full local system compromise, another massively popular router falls to the IoT botnets, even high-quality IoT devices have problems, the evolution of adblocking and countermeasures, an important update for Mozilla's Thunderbird, a bit of miscellany, listener feedback, and an update on the NSA's possible intervention into secure encryption standards.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-644.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-644-lq.mp3

---

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Our first show of the new year, and there's lot of little tidbits to clean up - including, yeah, a Macintosh zero-day, discovered just a couple of nights ago. It's all coming up next on Security Now!.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 644 for Tuesday, January 2nd, 2018: NSA Fingerprints.

It's time for Security Now!, the show where we cover your security and privacy online with the king of the tinfoil hat brigade. No, I'm not going to say that. He's our commander in chief, the man who protects us from all that is evil, Mr. Steve Gibson. Happy New Year, Steve.

**Steve Gibson:** Leo, great to be with you again, as always, for Episode 644.

**Leo:** Holy cow. Holy cow.

**Steve:** On this January 2nd of 2018. So, yeah, I think heading into another year of great interesting podcasts. And also...

**Leo:** Of course the good news is security is going to be tight, and we won't have to

do much this year. No? No?

**Steve:** You know, I was noticing how much shift there has been over to IoT stuff.

**Leo:** Yeah, yeah.

**Steve:** I mean, it really is, I mean, it's what everyone wants and is using. And it's just like sort of this brand new frontier of learning all the same lessons over again.

**Leo:** Well, it was an IoT Christmas for a lot of people; right? They got their Echoes and, you know, I mean, this is the year.

**Steve:** Well, yeah, and the problem is people really want the features. They want the convenience. They want to know if they left their garage door open as they're driving away. They want to be able to, well, I mean, Ring is a sponsor of the show, a major IoT device that allows you to be on the beach in the Bahamas and tell the delivery...

**Leo:** Which I was, and did. It's the funniest thing to be able to know who's at your door while I'm on a sailboat in the middle of the Caribbean. The doorbell was going off, and I said, hey, somebody's at home.

**Steve:** I can imagine, yes.

**Leo:** That's pretty funny, actually.

**Steve:** And in fact I was mentioning that Lorrie and I, my girlfriend and I went up to visit my sister and her family, and they've got a Ring on the front door, and it was going off from time to time as people were approaching.

**Leo:** Dong dong dong, yeah.

**Steve:** Yup, exactly, so that familiar sound.

**Leo:** So, now, this is the time of year when people come home from explaining to family various issues. And we were talking on MacBreak Weekly where everybody had explained the battery issue on the iPhone. And I imagine that you spend - do you spend time when you go home securing family computers and things like that?

**Steve:** No. My sister's husband is a techie.

**Leo:** Oh, good.

**Steve:** And so he's got that well in hand. And in fact, while my mom was alive, he was happily the one responsible, or that is to say, saddled with, you know, "Do I have WiFi?"

**Leo:** Yes, Mom, you do.

**Steve:** "Where is it?" And you know, it's okay, "Ken, go fix that." So, yeah, so I'm a safe distance away, although I do have that role down here in Southern California for a number of...

**Leo:** Yeah, of course you do, your friends, yeah.

**Steve:** ...friends and family and so forth. So we've got a lot to talk about, although I have to say, interesting topics, but not a huge number. Everyone was maybe enjoying the holidays.

**Leo:** Even hackers take the week off.

**Steve:** Even hackers, yes. The title for this week's podcast is "NSA Fingerprints," following an interesting posting by our friend Matthew Green at Johns Hopkins, a cryptographer. And I noted that his bio now says "Professor." And I'm pretty sure it used to say "Associate Professor."

**Leo:** Yes. I think he got tenure. I think you're right.

**Steve:** I think that happened, so congratulations to Matt. Anyway, we've talked in the past about this interesting oddity in one of RSA Security's random bit generators, a DRBG, Deterministic Random Bit Generator. And this was the Dual EC, the Dual Elliptic Curve generator, which was slower than the other three, and weird, and no one would choose it, but it was the default. And so that, you know - and this was years ago we discussed this, when this news came out.

Well, something just happened that surprised the cryptography community, that furthers the impression or provides additional evidence that - because there's another big piece of this that just kind of all came together as this weird side effect of our movement towards the next version of TLS, TLS v1.3, which really pretty much puts the last nail in the question of whether the NSA was actually up to some funny business. And remember that Reuters also reported back at the time that RSA had been paid $10 million to...

**Leo:** I remember that.

**Steve:** ...choose this funky and not clearly secure source of entropy for their crypto

library. But people were kind of thinking, eh, okay, well, so that's not good. Well, it turns out, yeah, it's actually pretty bad. So those pieces have come together, which we will end the podcast with.

But in the meantime, or until then, we're going to talk about a new and clever and disheartening abuse of our browsers' handy-dandy username and password autofill, which shouldn't be a surprise to us. Everybody is clever. And unfortunately all of this technology that we're using can be abused when people focus themselves on that. And so we have another instance of that which has been found in the wild being deployed by 1,110 of the top chunk of websites on the Internet. We'll talk about that.

Also there is something afoot. And for a while the initial reports were speculative. Now, in the last couple days, some additional information has come to light. There's been some frantic scurrying around by many OS kernel developers, changing some things that led the early people who were watching this to believe that there was an undisclosed major problem that was like that there had been nonpublic disclosure to kernel developers about. And now we think we know more about it, but it actually is going to have an impact on the performance of our systems. So we'll talk about that. There is a just-released macOS zero-day which allows full local system compromise.

**Leo:** What? We missed it.

**Steve:** Beautifully, beautifully written up.

**Leo:** Oh, boy.

**Steve:** I mean, the guys did a great job. There's also another massively popular router has fallen to the IoT botnets. Also I wanted to note that even high-quality IoT devices - Sonos - have problems on the public Internet. And I don't know if you've noticed yours updating more recently.

**Leo:** Lately, yeah, yeah, yeah. Just got an update.

**Steve:** Uh-huh. And we know why now.

**Leo:** Uh-oh.

**Steve:** So we'll talk about that. Also there's - I wanted to briefly touch on the evolution of adblocking and countermeasures to it. We've, of course, that's been a topic of ours because it revolves around privacy, and the technology is of interest and interesting. Well, this has been evolving. In a new study that'll be presented at a conference later this February, next month, has updated or matured, I guess, their impression of how much websites are responding. And it's much more than people knew. Also there's an important update needed for users of Mozilla's Thunderbird email client, which I just want to shake people up about a little bit. We have a little bit of miscellany, some listener feedback, and then we're going to talk about what recently came to light that leads us to believe that the NSA is a little more involved...

**Leo:** Uh-oh.

**Steve:** Well, it provides additional evidence of, I should say, the involvement that they've been suspected of until now.

**Leo:** All right.

**Steve:** So our picture this week you got a big kick out of...

**Leo:** Oh, I love it. I love it.

**Steve:** ...when it first came up on the screen, before we began recording. And this is just sort of the - I don't know how to describe it. It's some sort of a - it's something called the "output matrices" of apparently a major TV network. And it just sort of - it demonstrates the bailing wire and chewing gum approach to keeping something online. Apparently these things were overheating because they weren't in an adequately cooled environment. So somebody said, okay, well, we need to have a fan blowing on them. But of course the fan was too short, so they got a cardboard box and stuck the fan on top.

**Leo:** My favorite part is the electrical tape holding the fan to the...

**Steve:** Yeah, it's like duct tape-width black electrical tape holding the whole thing down. Oh, I guess it's sitting on another table that is resting on the floor. And then there's a sign on the front of this Rube Goldberg contraption saying, like explaining what is going on here.

**Leo:** What the heck?

**Steve:** And it says: "This fan is keeping the output matrices cool."

**Leo:** Oh.

**Steve:** Yeah, well, you know about those. You need your output matrices to be kept cool.

**Leo:** Always keep your matrix cool, yeah.

**Steve:** That's what I say. "If it is moved, all of the UKTV networks will fall off the air."

**Leo:** Can you read the - there's an asterisk and fine print, and I can't read it in this

image. Can you?

**Steve:** I can't either. I would love to know what that says. But, yeah. So, yeah, that's not a fan that you want to, like, bring over to your desk when you're too warm.

**Leo:** Don't move the fan.

**Steve:** Leave this alone. This is high-tech, yes, equipment plumbing. And, boy, if the UKTV networks are relying on this, then let's hope the fan keeps spinning.

Okay. So betraying our browsers' autofill. Some researchers, and they've been busy at the Princeton Center for Information Technology Policy. We've had a couple of stories about their publications and research. That's at the Freedom-to-Tinker.com, with hyphens, Freedom-to-Tinker.com site, where they blog about work that they're doing in privacy. They performed an extensive scan of the scripting being deployed by a bunch of the top Internet websites, and they discovered something disturbing.

They discovered that at least two commercial companies, one called Adthink at AudienceInsights.net - and Leo, my computer can't go there anymore because I don't want my computer to go there. But yours can, and our listeners can. Go to AudienceInsights, all one word, dot net. And you're going to find out what your ID on their network is. Mine is A008bc977d7aa23 blah blah blah. And everyone apparently has a unique ID because this company has been tracking us for quite some time, AudienceInsights.net.

So when you go there, they say: "What is it?" And they say: "AudienceInsights is a web application offering analysis and statistics based on anonymous data collection." Then they say: "How information is collected? Our partners' websites and applications host our program" - which is to say JavaScript - "that collects information and sends it to our servers where it is stored and analyzed." Okay. Then they happily tell me my own ID: "Your ID on our network is" and then that long string of looks like hex gibberish.

Then they ask themselves the question, "What is collected?" And they say: "We collect only anonymous data through anonymous cookies and technologies" - okay, that's the "and technologies" part - "that record, first, events related to your activity on the partner's website, such as the number of pages viewed or your searches made on the partner's website." And I'll note that anyone who cares who's running a web server can easily determine that for themselves. So no third party is necessary. And, second: "Information provided by trusted partners that may include sociodemographic data such as age range." Well, that's sort of the least of what they apparently collect because the Princeton researchers discovered, when they looked more closely at this code, that birth date, age, gender, nationality, height, weight, body mass index...

**Leo:** Wait a minute. Where are they getting this from?

**Steve:** Apparently this is the data contributed by their trusted partners.

**Leo:** Jesus.

**Steve:** Eye color; net income; education; occupation; raw income; relationship states; seek for gender M, F, transman, transwoman, or couple; pets; location with postal code, town state, and country; loan, loan type, amount, duration, and whether you're over indebted; insurance for car, motorbike, home, pet, health, and life.

**Leo:** Wow.

**Steve:** Card risk, including chargeback and fraud attempt. Whether you have a car and, if so, make, model, type, registration, model year and fuel type. I guess they want to know if it's electric...

**Leo:** How do they know all this?

**Steve:** Also tobacco, alcohol, travel - from, to, departure, and return. And I love this, your car hire driver age. Oh, and hotel stars. Those are the categories which this script is apparently filtering through and sorting. Then they say: "We do not collect any personal information. We do not know who you are. We do not know your residential address."

**Leo:** We know what kind of fuel your car uses.

**Steve:** They know what type of, apparently, what your sexual preference is for people you are seeking.

**Leo:** Geez.

**Steve:** And so on. So what they don't say in all of this happy page - and again, it's go to AudienceInsights.net, and you can learn your ID on their network. What they don't say is that they deploy - get this, Leo - hidden scripts running on these trusted partners' web pages which deliberately trick the users' browsers auto form fill into populating the user's login username and password for that website.

**Leo:** Yeah.

**Steve:** Which then, after being hashed, is sent back to Adthink's server farm for collection. So first of all, the data is hashed, so they're not obtaining - that is, the script is choosing to hash it. If the script chose not to hash it, or if because that data is being sent along with a cookie from the browser, they're able to link all this together. And also, since the problem with hashing is that their - and their whole goal is this is a non-cookie form of tracking.

But what the Princeton researchers found disturbing, aside from the breadth of

information which they're hoping to collect from people, is that this is arguably abusing the browser's willingness to populate username and password fields on behalf of its user. So by hiding this browser autofill, they are getting hashes of the username, which are in many cases the user's email address, which gives them cross-domain tracking because they'll get the same hash, and the user's password, which, if they're not using a separate password for every site they visit, will again create hash collisions, so they'll know that this user has popped up on a different site.

So anyway, there are two domains. And the reason I said you need to go there and I can't is I immediately put static.audienceinsights.net into my system's hosts file, aiming it at 127.0.0.1, and api.behavioralengine.com. I have all of this in the show notes, for anyone who's interested. Those are the two companies that are doing this that the Princeton researchers poking into what current scripting behavior was causing passwords to be populated on invisible form fields. And I just, I mean, a quick hack is just to route those two localhosts so that our own browser is unable to initiate a connection to them in order for this information to get populated.

So what the researchers wrote was: "We found two scripts using this technique to extract email addresses from login managers on the websites which embed them. These addresses are then hashed and sent to one or more third-party servers. These scripts were present on 1,110 of Alexa's top one million sites." So, okay, 0.1%, not a high percentage, but a thousand of them, more than a thousand of the top million.

"The process of detecting these scripts is described in our measurement methodology," they wrote, and I've got a link to their full PDF. Also, at webtransparency.cs.princeton.edu is a complete list of all the sites which are performing this autofill behavior. And there is over on GitHub and also links here in the show notes a list maintained that looks like it's set up to drive pfSense for blocking any of these sites. The good news is I did a cursory scroll through, and I didn't see any major sites. This looks a little more like some of the more fringe sites are doing this. But unfortunately this is an instance where any of these so-called "trusted partners," I mean, they're not partners we trust, they're basically the other sites that these two ad companies are in cahoots with in order to build a database that they're able to somehow leverage for more profit.

Oh, and there's also a demo page that I have a link to in the show notes where you can go, if you're curious, to actually watch this script approach being leveraged. And so the takeaway is, first of all, you may want to just blackhole these two domains static.audienceinsights.net and api.behavioralengine.com if you object to this behavior. But this really begs the question, the obvious solution is for browsers not to be so quick to autofill. That is, maybe for them to be smart about whether the fields being shown are visible to the user. They should certainly be able to do that.

And I remember you and I played with this whole autofill problem sort of in a different instance months ago, where you discovered that just the browser's information about us, which it has available to helpfully populate form fields, could be brought up offscreen, like at a negative 1,000 coordinate of the screen, and not be visible.

**Leo:** And this is also tied to the fact that we learned that you don't have to press the submit button for these fields to be read; right?

**Steve:** Correct.

**Leo:** That these fields are available to the browser and the site without submit.

**Steve:** Exactly. And in fact there's an on-change event which, for example, I use over in the Password Haystacks, where, think about it, as the user is typing, I'm updating the calculations about alphabet size and time required to brute-force the password on a character-by-character basis. So there's an on-change event which a field can be given such that, when its contents changes, some script is notified. So the script would just sit there. And when the browser populates the field with the information, that fires off the on-change event. The JavaScript wakes up, responds to the event, hashes the data, and sends it off to the mothership.

And so, yeah. So as you said, Leo, the first problem is that this is done before you hit Submit. And of course from a UI standpoint we probably need that to happen. The user needs to know whether their browser is going to populate the field or not. But it would be nice if, I mean, the only mitigation anyone has suggested, and there's been a lot of dialogue about this, is to not make it automatic. Require some user interaction in order to cause the browser to fill the fields, but then of course that breaks the user experience. It's so handy and convenient to have this happen.

So really what I hope is that this will get enough attention that browser vendors will realize this convenience, and it's really their responsibility to the user to do the right thing with the information the user has entrusted to the browser, is being abused, and to come up with a way to block these scripts from being able to collect username and password data offscreen because note that these guys are hashing this and not taking advantage of it. But if anybody else gets their script on someone's site that is able to do this, well, they're collecting usernames and passwords.

**Leo:** Right.

**Steve:** Which is not a good thing.

**Leo:** These are first-party scripts, though; right? So an adblocker probably wouldn't block them.

**Steve:** That's probably true.

**Leo:** [Crosstalk] asking if Quad 9 would block them. I think if it's a first-party script they can't.

**Steve:** No. Now, Quad 9 could blacklist those domains in order to prevent them, in order to prevent your...

**Leo:** From sending information back.

**Steve:** Yes, yeah.

**Leo:** Okay. So that wouldn't be - that's, yeah. So sending it to somebody who is a third party.

**Steve:** Right, right. So again, as I was saying, I think we're going to have to see if browsers come up with a way of blocking this behavior. And once again, a feature that was intended to help us and to remain between the site we're visiting and our browser has been compromised by companies saying to those sites, hey, put some of our script on your server so that, as you said, Leo, it becomes first-party script and then has access to everything that that site would normally have access to. I mean, so this really is, it's the fault of those websites hosting script which is sending their users' usernames and passwords to a third-party service, and presumably through some sort of financial arrangement. So creepy behavior, but nothing we can do unless our browsers behave themselves.

Okay. So we've talked about ASLR years ago, Address Space Layout Randomization, where all contemporary OSes today, because this is a problem, are deliberately rearranging where things end up residing in memory. In 32-bit OSes we have potentially a 4.3GB address space. In Windows, about half of that is lost to the OS. But even so, the user space is still two billion bytes. So the good news is most apps are not that big. So there's enough empty space that chunks of applications and libraries that they're loading and pieces of the OS that run in user space can all be scrambled around in memory. That's important because, if a buffer overflow is found, the randomization of where things are makes it significantly more difficult for that overflow to be exploited.

Once upon a time the overflow which often occurs on a dynamically allocated buffer on the stack, the stack itself could be used to contain executable code. Well, then we fixed that by making the stack nonexecutable with the so-called NX bits on various platforms. Well, so that meant the hackers had to work harder. They couldn't bring their own code along with the overflow. But they very cleverly realized, oh, we can just use existing code which is executable, in so-called Return-Oriented Programming, ROP, where they would jump to some code at the end of a subroutine that got a little bit of the work they wanted to get done, done for them.

Then it would come back, and they would jump again somewhere else and stitch a bunch of those little pieces together into performing some feat that they needed. And sometimes it's as simple as, like, flipping off a protection bit. For example, if you could find some code somewhere which, although it probably wasn't intended to, turned off the no-execute bit for the stack, suddenly your stack becomes executable. So sometimes you just need to flip the right bit somewhere.

So that same technology then moved into the kernel to give us KASLR, which is Kernel Address Space Layout Randomization, to provide the same features because, naturally, people began abusing kernel code if it wasn't also being jumbled around.

Okay. So we had Address Space Layout Randomization for the user, Kernel Address Space Layout Randomization for the kernel. A couple months ago, some signs of a breakthrough in hacking Kernel Address Space Layout Randomization began to surface. And this probably occurred initially very much on the QT in the summer of last year because it was subsequently observed that Windows kernel began getting some patches, and a very short turnaround on the Linux kernel development side was witnessed and caused a lot of people to wonder what was going on.

A new technology was created known as Kernel Page Table Isolation, KPTI, which is a

hardening technique, probably first adopted in the Linux kernel to improve security by better isolating the user space and kernel space memory. We've talked in the past, when we were toward the end of our series on processor architecture about virtual memory and the way that physical memory, which starts at zero and runs up to however much physical memory you have, is seen in the abstract by not only the kernel itself, the operating system, but most specifically by user processes.

So, for example, every user process thinks it's starting with a very similar snapshot of memory. For example, it might see memory beginning at 40,000 in hex, that is, every process sees sort of the way the OS wants to present memory to the process the same. But they can't be actually seeing the same physical memory. First of all, you need the processes to be isolated so they're not able just to poke in each other's memory, but there's a level of indirection.

So the idea is that the memory addresses which processes use is looked up through a series of page tables. And on the Intel architecture there's like a chain of three of them. So the page that the memory access that a processor makes with typically 4K bytes granularity is looked up to see where it should, that is, that page is virtual memory. It's looked up to see where it actually is in a first table; and that table, the contents of that table is looked up by a second table; and the contents of that table is looked up by a third table. And so it's unbelievably complicated. It's very powerful and very flexible.

But the only way this can possibly work is if these tables are cached; if they are kept in high-speed, on-chip, locally accessible memory so that all of these lookups can happen fast. And there's something know as the TLB, the Translation Lookaside Buffer is the term, which also even further caches the past results of this multilevel lookup. Well, there's been a shortcut that kernel developers, all kernel developers have been taking for years because it was understood that having the kernel and the user space sharing these cached tables was a theoretical problem.

Intel implemented a sort of a mitigation to this problem known as Process Context Identifiers, PCIDs, like 10 years ago. So they've been available in chips for a long time. But the problem is it creates a performance hit. If what you really want is code running in user space, any of our processes, to be able to call down to the kernel, whether it's Linux or Windows or macOS, you want it to be able to call down to the kernel as quickly as possible. I mean, the applications are making API calls, kernel API calls, all the time.

Now, there is a lot of OS now running in user space; but still, for interrupt handling and device driver and I/O things, that's all being done - and memory management - that's all being done by the kernel, where it has to be in order to have the isolation required. So all of the OSes have for years been sharing the page tables used by the kernel with the page tables being used by applications, purely for efficiency's sake, with some little bit of anxiety that maybe there would be a problem found in this.

Well, what apparently happened a few months ago is a problem was indeed found. And over the last couple weeks, where there's been a lot more churn among the developer community about what's going on, there's been additional information which has surfaced. For example, in the Linux kernel where this work has been done, comments have been redacted, and the documentation pages for this work are absent, apparently because they want to get these mitigations in place before this thing gets more public.

Interestingly, it looks like this is purely an Intel architecture problem. It does not affect AMD processors. So what this Kernel Page Table Isolation does, KPTI, is it reluctantly separates the kernel page tables from the user space page tables at some significant cost in performance. It looks like operating systems which implement this are going to take

something like a 5% performance hit across the board because what is going to probably emerge soon is that it has never really been safe to do this, and that some clever developers or hackers really have figured out how to break the isolation which sharing page tables with the kernel has, for the sake of performance, has been providing.

A kernel developer by the name of Tom Lendacky said: "AMD processors are" - he posted to a list recently. "AMD processors are not subject to the types of attacks that the kernel page table isolation feature protects against." He wrote: "The AMD microarchitecture does not allow memory references, including speculative references, that access higher privileged data when running in a lesser privileged mode when that access would result in a page fault."

So that gives us a big clue. That, if you read that carefully, that's suggesting that something about the Intel architecture's speculative accesses, remember that that's the way Intel has improved performance, is that they're doing what are known as speculative accesses, where the CPU is running ahead of the code in order to prefetch things that the code may need. Apparently, they're performing speculative references of data that would result in a page fault, that is, that the actual process has no legitimate access to; and that speculative reference is altering the cache, the contents of the caching in the chip, which we know that precise timing analysis can reveal cache hits and misses with sufficient granularity and resolution to cause a security problem.

Anyway, so what Tom wrote in his posting was "Disable page table isolation by default on AMD processors by not setting the" - and then here it is - "X86_BUG_CPU_INSECURE feature." So there is a new feature flag, X86_BUG_CPU_INSECURE, which controls whether this page table isolation is being used. And then in some other digging that I did, I found a tweet from @pwnallthethings. Matt Tait on Twitter said one possible vector is if speculative operations can influence what the processor does with the cache, the results can be observed with cache timing attacks. If the branch predictor reads the results of a speculative operation, it's quite easy.

So anyway, there's been a lot of people wondering what is happening and what's been going on. And finally, in a post on Tumblr, a guy posting as Python Sweetness said in his TL;DR at the top of a much longer posting, he said: "There is presently an embargoed security bug impacting apparently all contemporary CPU architectures that implement virtual memory" - which, yes, as we know, is all of them - he says, "requiring hardware changes to fully resolve. Urgent development of a software mitigation is being done in the open and recently landed in the Linux kernel, and a similar mitigation began appearing in NT" - and by that he means Windows - "kernels in November. In the worst case, the software fix causes" - he wrote "huge," I'm not sure that 5% qualifies as huge, but still significant - "slowdowns in typical workloads. There are hints the attack impacts common virtualization environments including Amazon EC2 and Google's Compute Engine, and additional hints the exact attack may involve a new variant of Rowhammer."

Well, now, this doesn't actually look like Rowhammer based on information that has come to light subsequently. But it does look like probably in the next week or two, once this gets pushed out and implemented. It's just landed in - I don't have it in front of me - one of the most recent Linux kernel builds, and it's being moved back then into 4.15. So it doesn't look like anything is happening yet over in the FreeBSD source tree. But it does, it looks like something interesting is impacting other kernels.

So we'll stay tuned. If this is the case, if this is exploitable, it means that our OSes are going to have to isolate user space and kernel space page tables, which will give us some sort of a performance hit. AMD-based systems don't have to do it. It looks like it's a glitch. I mean, a clever something that Intel did to squeeze the last drop of performance

out of their execution pipeline speculation, and that somebody has found a clever way of leveraging that into an information disclosure about the contents of the page tables which, if used, would allow a bypass of the kernel ASLR and then access to kernel code again. So, wow, I mean, this is - we're really down in the weeds with this stuff now. But we have very complex systems. And as we keep saying on the podcast, security and complexity are archenemies of each other.

I mentioned a zero-day attack which was just published. A Swiss hacker who goes by the name Siguza, S-I-G-U-Z-A, and who describes himself as a hobbyist developer and hacker, dropped a tweet on New Year's Eve. And he said: "Eff it." And he didn't say "eff," he spelled it all the way out. He said: "Dropping a macOS zero-day. Happy New Year, everyone." And then in his tweet was a link to his page on GitHub. He has his own subdomain at siguza.github.io. He called this "IOHIDeous," I-O-H-I-D-E-O-U-S. And HID, of course, is Human Interface Device. He claims, and it's pretty well substantiated now, that this is a more than 15-year-old bug which has long existed in the Mac operating system family.

He provided a proof of concept which targets all macOS versions, crashes the kernel to prove the existence of a memory corruption. He has what he calls a "leak," which targets High Sierra, just to prove that no separate KASLR leak is needed. And the third piece of code is HID, which targets Sierra and High Sierra. He says: "Up to 10.13.1," and then points people to the readme, "achieving full kernel read/write and disables SIP" - which is the System Integrity Protection which Apple added not long ago - "to prove that the vulnerability can be exploited by any unprivileged user on all recent versions of macOS."

So this is not a remote execution, but this is a full-kernel read/write compromise from any unprivileged user on all recent versions of macOS. So that means, if you run some code on your machine which you should not trust as a, well, which you should be able to trust as an unprivileged user, relying on the OS to prevent code you didn't write from abusing the inherent trust. That code can do pretty much anything it wants to on your system. So I didn't dig into this in depth, but I did look at it closely enough to be very impressed with this hacker. I mean, this guy really knows his stuff. And there is absolutely no question in my mind that this is the real deal, and legitimate.

In digging around more, I found that he had jumped into a dialogue about this over on YCombinator, where he wrote, he said: "I had actually submitted to the ZDI [Zero Day Initiative], but I had written the exploit and write-up in the first place mainly because," he wrote, "I like hacking rather than for money. I figured I'd see what offers I'd get anyway. But once I had spent all the time on the write-up, I mainly wanted people to see that, and the amount offered wasn't enough to convince me otherwise."

He wrote: "I might've published this earlier even, but my December was kind of busy, first with the v0rtex exploit and then with," he wrote, "34C3," which of course is the Chaos Convention, which has just happened. He said: "And an engineer from Apple's security team contacted me a bit after releasing. They had found the bug a while ago," so this wasn't news to them. I'm editorializing. It wasn't news to them. He wrote: "...but hadn't verified the subsequent patch which actually did not fix it. And a while ago I tweeted this," and he has a link to some interaction with them. "So they do have people on it," "they" meaning Apple. He said: "I also told that person to extend my condolences to whoever has to come in and fix that now. But they basically said that there's nothing to apologize for and that they, the team, really like such write-ups."

And I don't blame them. I mean, this guy was super thorough so that, I mean, he told them exactly - he told them, Apple, exactly what they need to do to fix this. He said: "So I guess I'm not that evil." And by the way, this was responding to people on YCombinator

saying who is this guy dropping a bad Mac zero-day on New Year's Eve? And he said: "And I neither want to watch the world burn," again referring to one of the earlier comments, "nor did anyone brush me the wrong way," another comment.

"I didn't publish this out of hate, but out of love for hacking. If you're concerned about skids hacking you now, they need to get code execution first on your machine. If you're concerned about people who can do that, then those can also get kernel read/write without me, so nothing really changed for the average user." And then people were wondering whether this was actually he, and he said: "Yes, it's really me. Will add keybase proof if my karma gets above greater than two," which did happen, and then so he did provide some proof. But anyway, there is a...

**Leo:** Plays reddit like a fiddle. Man.

**Steve:** Yes. So there is a problem. Apple knows about it. They've been working on it. While they would have probably liked to have this communication in private, they are grateful for the communication. And if anyone is curious, I mean, it is an amazing piece of beautiful detailed work which will allow Apple actually to get this fixed properly, probably much more quickly. So look for a macOS update in the very near future because this is bad. This is a zero-day which has only been known about for a few days. And so it's zero-day plus three or plus two, and people are going to want to get their macOS patched.

**Leo:** I hope this doesn't become a trend, though, of people just not following procedure and just releasing stuff because he sounds kind of like an ass. Obviously he's a smart kid.

**Steve:** Oh, Leo. When you see the - if you scroll through this work he did, I mean, it's eye-popping. I mean, the guy really knows his stuff.

**Leo:** But just, you know, tell Apple. He's so anxious to get the credit that he didn't want to tell Apple ahead of time.

**Steve:** Yup. Yup.

**Leo:** But to reiterate, somebody has to have code execute access to your machine. That means not necessarily physical access; right? If there were another hack to allow them to execute code...

**Steve:** Well, no, if they just post a malicious app somewhere.

**Leo:** Right.

**Steve:** I mean, so an unprivileged user...

**Leo:** Can escalate.

**Steve:** Should be, yes, should be protected by the OS.

**Leo:** Right.

**Steve:** That's why you have to have root login - well, now - on the Mac in order to do privileged things. This allows non-root code that you would run by mistake to do whatever you could as a root...

**Leo:** So somebody'd have to put this in a malicious app. Or it could be a website; right? I mean, a website could probably do this if it had malicious code on it; right?

**Steve:** Yeah. They probably, you know, it would be like, oh, here, download this, and you'll be glad you did.

**Leo:** Yeah, he's going to do that, yeah, yeah. He said he spent 250 hours on this. But again, more interested in the glory than anything else. And I think that's too bad.

**Steve:** Yet another IoT device falls. The code used in a zero-day to attack Huawei routers has been made public. It was used by one of several Mirai botnet variants. Remember last week or two weeks ago we were talking about how there are unfortunately, because the Mirai source code got posted, lots of Mirai botnet clones have sprung up. But apparently those clones are also being inventive. They're not all just following off of Mirai because one known as Satori was found to be compromising Huawei routers. In fact, it compromised hundreds of thousands of publicly exposed Huawei routers over the past several weeks. And of course the problem is routers are supposed to be publicly exposed. They're not supposed to have publicly facing vulnerabilities, but the router by nature is WAN-facing, so it's out there. And so thus its security is really crucial.

The exploit code for compromising Huawei routers was recently posted to Pastebin and is now public. So researchers who have been watching all of this now expect that the code will quickly become a commodity and be leveraged in future DDoS attacks via other existing botnets such as Reaper, which is still going strong, or IoTrooper. And I also saw some reference to Brickerbot in this context.

So two weeks ago Check Point Security identified the vulnerability. And I have to say we were at the end of 2017, and my eyes sort of popped when I saw the CVE number because the CVE, the common vulnerabilities registration, always starts at zero at the beginning of the year. So that was actually one theory of why that other guy posted his New Year's Eve macOS zero-day was that he wanted to score CVE-2018-0. I don't think he did. But my eyes popped because the Check Point CVE about the zero-day Huawei router is CVE-2017, of course, for last year, hyphen - and get this: 17,215. So, yes, 17,215 vulnerabilities logged in 2017. So it'll be interesting to see how 2018 fares by comparison. And Leo, again, as you said at the top of the show, I have a feeling there

will be more in 2018 than fewer than we saw in 2017.

**Leo:** You ain't seen nothin' yet, boys and girls.

**Steve:** 17,215. I don't know what the actual last number was of the year, but this one would have been recent. Anyway, they discovered, in this case it was a Huawei home router Model HG532, that it was being exploited to spread Mirai, that Mirai variant, the Satori variant. Since then, Huawei issued an updated security notice to customers warning that the flaw allows a remote adversary to send malicious packets to port 37215 to execute remote code on vulnerable routers. Now, of course, remember, the good news is mostly people just want these routers in order to commandeer the bandwidth of their owners. But if somebody can hook to your router on port 37215 and execute malicious, like random code on it, then they've got access to your network, which you really don't want.

So if you don't have - if you have a router like this which is compromised and not fixed, it's another real good reason for maybe sticking another router inboard so that that router talks to your router, which then talks to your network. That way they're not able to get any - if they've compromised your WAN-facing router, they cannot get in past your second router because it's hopefully behaving itself the way it should. And certainly it's not easy for them to scan it because, well, I guess they could scan it if they really were able to run arbitrary code on the outboard router.

Anyway, I will just say get a good router, one that's being maintained. And these guys, I mean, Huawei responding as quickly as they did is behavior you would want. The problem is these are hundreds of thousands of consumer routers. And consumers are, like, connected to the Internet, and they're happy. They don't realize that their router has been compromised behind their back.

Anyway, so if you happen to have a Huawei router, do go to Huawei and make sure you are running the latest firmware because hundreds of thousands of these routers are being compromised, and it looks like what's happening is researchers are expecting that with this new exploit going public, essentially, all the other Mirai variants will jump on it and add that to their bag of tricks.

So even high-quality IoT devices can have problems. The good news is our favorite Internet-connected or network-connected speakers, Leo, our Sonoses...

**Leo:** Yes?

**Steve:** ...were not remotely compromisable. Or maybe I shouldn't state that. The Trend Micro researchers who poked at both the Sonos and at the Bose networked speakers found a worrisome array of problems in both. But what was more worrisome is that somewhere between 4,000 and 5,000 Sonoses are publicly exposed to the Internet. And you just have to wonder how that is happening. I'm tongue-tied. I was going to say this is further reason why I think that, if anyone needs to do a New Year's Resolution, it is logically segment your LAN so that IoT devices, where practical, can be on their own network.

The reason I say that is that people probably want their smartphone and their tablets to be on their secure network, except that, if you put a Sonos or other IoT system on a

segmented network, well, by definition you cannot access it. You can't get to it. So it does create inconvenience if the devices you're trying to have in the maximum security network are unable to communicate with, but in order to use the IoT devices they need to. I don't have a solution for that except, for example, if it would be feasible, maybe use an older iPhone or older smartphone, Android, to be the controller of your IoT things, if that's feasible. It lives over in the insecure network, and you remember not to do anything important on it because it's over in the insecure network. And you keep your other devices more safe, or at least your PCs, where you probably don't need the PC to be talking to IoT devices if your, for example, your smartphones can. Leave all of them over on the segmented network.

Anyway, the point is we keep running across, and I'm sure this trend is going to continue and accelerate, as you said, Leo, this is the IoT Christmas because so much has happened in the last year, people were giving each other all kinds of Internet-connected things.

So Sonos responded very quickly to the Trend Micro researchers' reports, as quickly as anyone could ask. They just chose Sonos because it was popular. They also looked at Bose, although there was like four times, four or five times more Sonos systems that were just available due to Sonos's popularity. What they found was, believe it or not, an internal web server exposed to the public Internet. Shodan could be used to search for them and revealed more than 4,000. And a subsequent search showed 5,000, but they realized there may be some IP overlap between them because IPs on consumer systems don't tend to be rigidly fixed. But at least, in a single scan, more than 4,000 Sonos exposed web servers.

So our little cute Sonos speakers, which are very popular because the quality is very good, and then the feature set is nice, I really like mine. And I am glad that I'm running them on a segmented network because experience demonstrates we have to. The problem is for some reason, and I haven't yet gone to look because I wasn't worried about this from a security standpoint, but now I'm curious from an intellectual standpoint, do these all use UPnP to map themselves for public access? It would be disappointing if they did that deliberately. Maybe these are just somehow misconfigured or deliberately configured for that mode of operation, like they for some reason statically mapped the WAN into their Sonos device.

But for whatever reason, there are more than 4,000 Sonos speaker web servers visible on the 'Net. And I think I had here which port they were using, but I'm not seeing it in my notes. There is an extensive PDF that I've got links to from the show notes, which also shows a map of a big chunk of Europe, I meant to put it in the show notes and forgot, showing like their own geographic map of Europe just covered with Sonos speaker installations, each of which has an exposed web server. So through the web server you are able to do a bunch of things. You are able to look at the device's ARP table to discover the physical MAC addresses of all other devices on the LAN which the Sonos speaker is sharing with the LAN. The most worrisome feature is you can look at the logon credentials of any music services which you have registered with the Sonos because of course it has to log...

**Leo:** Oooh.

**Steve:** Yes, yes, not good. That's been fixed, so that's no longer flapping in the breeze, but it was until Trend Micro said, uh, you know, we could see our Spotify account information.

**Leo:** That's not good. Password in the clear?

**Steve:** Yeah.

**Leo:** Wow.

**Steve:** Yeah. Because, again, the device needs it. It needs to send it in the clear in order to log onto Spotify and wherever, Pandora and so forth, in order to access that music content.

**Leo:** Yikes.

**Steve:** Yeah. Also there's a very handy network page which allows you to issue your own network commands, like ping. And so you're able to do a broadcast ping and get affirmative responses from every pingable device on the LAN. So the point is, while this isn't a huge security problem, it really is, or was, a significant information disclosure problem. And of less concern, but still a little annoying, they could peruse your music library, determine what your musical tastes were, determine what song you were listening to at the moment, if any, and basically get full access to what you're doing with Sonos. In extrapolating the potential nefarious purposes this could be put to, the researchers said, well, you know, for a targeted attack, if somebody wanted to get you, and knew you were a Sonos user, they could get into your Sonos and remotely determine what hours of the day you are using it and when you're not, and to some degree fuzzily infer whether you are probably home at given times or not.

So anyway, it's not a huge issue, but it's an example of how, I mean, even very high-quality devices. And I've had mine plugged in since the late summer, and I've been - they've been updating themselves a number of times over the last few months. And the first time I've installed any of those devices, they've done a firmware update. So the company is being responsible, but they must have assumed that this information, this access to the web server, was going to be restricted to other devices on the LAN, that is, never intending for it to be made public.

And that brings me back to something which is a constant issue with me, and an annoyance. And that is, I have never seen anybody ever, except me on this podcast, say or observe that all that has to be done by any of these devices where you really do not want remote access, is set the TTL, the Time to Live, of the packets that are emitted from the device to one or two. I mean, one. Remember that any router that encounters one of these packets will decrement the TTL and drop it like a hot potato if it goes to zero. That is a rule that is universally, I mean, it's like in an Internet, a global Internet where very few rules are actually universally followed, that's one that is. You have to honor that, or we end up with the Internet having packets that never die, that just circulate and spin around on the 'Net forever.

So every router decrements TTL. When it hits zero, it drops it and probably sends a notification back saying "expired." And we know that this is, for example, how the traceroute command is able to figure out how packets travel, by deliberately setting short TTLs which cause an expiration ICMP message to be sent back. My point is that, if devices like the Sonos that should never be accessible publicly simply emitted their

packets with a TTL of one or maybe two, just to be safe, then you could never access them remotely.

In order to run a web server you've got to set up a TCP connection. In order for that to happen, you have to have a three-way handshake, which means that you need to send them a packet, like it a packet, "it" the Sonos, and it needs to get one back to you. And so all of this is TCP, which if you emit TTLs of one, those packets die at the first router they encounter, either at your border, you know, the little consumer router may not decrement TTL. I'm not sure whether it does, or whether they all do. But certainly the first router that you hit at your ISP is going to decrement that and just say, sorry, we don't know what you're doing. You may be trying to traceroute us. But we're not forwarding that packet.

If people would do that, people who do not want their local stuff to be publicly available, it just ends it. I mean, you could have all your ports open. You could do anything you wanted, if you simply set TTLs to one. I've never seen it done, and I've never seen it talked about or mentioned.

**Leo:** This is not something you could do. This is something that the device would have to do.

**Steve:** Correct, correct. It's down in the stack. So it's the kind of thing you could do with raw sockets. But it's also something that someone building, I mean, these are all Linux-based. You just rebuild the kernel so the default TTL - and normally there's a setting because remember we've talked about this problem. Once upon a time TTLs were, like, 32. And it's considered the diameter of the Internet, that is, what are the two points on the Internet where there are the greatest number of hops.

And at some point, years ago, OSes were setting their TTL to 32 because, well, that's high enough. Well, no. There were so many routers added that the diameter of the Internet became greater than 32, and there were some locations you could not reach with a TTL of 32 because the packet would expire before it got there. So then we had to bump it up to 64. In some cases now it's 128. But it is normally something you can set. And so if they'd just set the TTL to one, this problem would never occur. You could connect to these devices locally with no problem at all because it's just device to device. But the moment it tried to cross a router boundary, that packet would die.

And that company, as a consequence, would have amazing security because it wouldn't matter if they left ports open, or they had insecurities that might end up going to be publicly facing on the Internet. It just - everything would just die. So I don't think of it often, but I talk about this whenever it occurs to me because it's a great security measure that we're completely missing, that I've just never seen anyone implement.

**Leo:** You've said this before, have you?

**Steve:** Yeah.

**Leo:** Wow. This is a good - seems obvious and simple.

**Steve:** It'd just be - and bulletproof. And like it cannot be defeated. It's just bulletproof.

**Leo:** Wow. It just doesn't pop out of your own network. It can't.

**Steve:** And you wouldn't think I'd need any more caffeine after that, but...

**Leo:** Well, too late. Go ahead, we're going to watch you drink. We've done all the ads. We shall enjoy the sight of Steve consuming beverages from his massive mug. Dilly dilly. Just say "dilly dilly." Go ahead. You don't know what dilly dilly is.

**Steve:** Leo, you don't need any more caffeine.

**Leo:** I've been hanging out with the teenagers.

**Steve:** That's right. So TechCrunch covered some interesting research which will be presented at the upcoming Network and Distributed Systems Security Symposium at the end of next month, end of February 2018. This was conducted by the University of Iowa and UC Riverside. They carefully profiled the behavior of the top 10,000 sites on the 'Net. And what they discovered was one third of the top 10,000 were showing some active awareness, concern, or in some cases deliberate anti-adblocking effort. That is, they were adblocking aware. Their behavior was different if a client did not follow through and pull ads from the site. And in some cases they were going to anti-adblocking efforts to thwart the efforts of ad blockers to block ads.

And it's interesting is that the top 1,000 sites, so the top 10th of the top 10,000, the top 1,000 sites had an even higher instance. They were at 38.2% of those top 1,000 sites altering their behavior in some way based upon the blocking behavior of the browser. So essentially they are seeing evidence of what we anticipated when we first started talking about advertising and adblocking, which is an escalation of this battle. Sites are beginning to decide that they need to show people ads, even when the people, their visitors have said, "No, I don't want ads here"; or in some cases have responded to a "please accept ads" in some fashion because we're ad...

**Leo:** I get that more and more now, yeah.

**Steve:** Yes.

**Leo:** I do that. I think we want - I want to support these sites.

**Steve:** I do, too. And what I'm wishing is that it was easier to do that. You know, we're all in a hurry. We're visiting somewhere quickly. We're not sure if there's anything here that we need. But unfortunately it's too difficult in several cases to whitelist a site whose ads you do want. And so I'm wishing...

**Leo:** In uBlock Origin you just go to the uBlock Origin dropdown and click that button; right?

**Steve:** And press the button.

**Leo:** It's permanent; right? Yeah.

**Steve:** That should be permanent.

**Leo:** It's not too bad, yeah.

**Steve:** I think it's whether it's that page or that site that I can't remember. So, yeah, you're right. It's not too bad.

**Leo:** I do it all the time. I think it's...

**Steve:** And worth doing.

**Leo:** Yeah.

**Steve:** Yes, and boy, Jimmy sure dunned me on wanting money for Wikipedia this last time.

**Leo:** Yeah, that's really annoying.

**Steve:** It really is. I gave him $100 and said go away. And then I kept getting, I mean, it's like they didn't know who I was.

**Leo:** Yeah, I'm thinking they probably don't want to set a cookie. Maybe they're just being privacy focused. You see? There's advantages to cookies. I give him money every month, and I still get that, you know.

**Steve:** Yeah. It's just like, okay, I mean, but they were pushing it this last season.

**Leo:** Well, they did this last year, too.

**Steve:** Oh, they do. Some of the information that I was just talking about in this podcast about the behavior of the page table stuff I got by doing a quick read through what Wikipedia had to say about it. So super useful.

**Leo:** Right, right. It's a very good resource, yeah.

**Steve:** Yeah. So a quickie for users of Thunderbird, if there are five of you out there using Thunderbird. I had no idea that the adoption was so skinny. I found a number of 0.7%. So I don't know how many people are using it. But for what it's worth, there was a bunch of significant problems just fixed, which are fixed in 52.5.2, which is the one that you want to make sure you're using.

I mean, again, it's hard to imagine with the adoption level so low that anyone would take issue. But email typically carries in its headers the identity of the email client. So somebody would know that you were a Thunderbird client user and could target you if they knew that and you had not patched. So just a quick heads-up that, if you're a Thunderbird user, make sure you've got the latest. Mozilla patched these problems quickly. There was a critical buffer overflow. It does allow an attacker's code to be run on your system. So make sure you've got that patched.

Also, I did want to mention, because I've been on this podcast bitching about how unusable my iPhone 6 Plus had become, and saying that, you know...

**Leo:** Now we know why.

**Steve:** Exactly. Exactly. Apple finally admitted to what we all pretty much knew or presumed, which is that the newer iOS releases were for some reason apparently deliberately underclocking the older phones. Well, now we know that's absolutely the case. And this whole thing seems a little fishy to me, Leo, even so.

**Leo:** Interesting. Okay.

**Steve:** Yes. I really baby my battery. And it's 100% strong, and it's a 6 Plus. So I skipped over the 7 and the 8, and I've stayed with the 6 Plus.

**Leo:** 6s or a 6?

**Steve:** 6 Plus.

**Leo:** There's 6 Plus or a 6s Plus. So you're saying it's a 6. Wow.

**Steve:** Yes.

**Leo:** That's pretty old, Steven.

**Steve:** I know. And, I mean, absolutely the battery, I can have it out all day, and it's at 95%.

**Leo:** You can check. There are battery tools. In fact, Apple's going to put it in iOS. It'll tell you how much of your original battery life is still available. It's obviously thinking, you know, we don't know what Apple [crosstalk].

**Steve:** Well, I will absolutely be taking advantage of the $29 discounted battery replacement for that phone. I have a 10. But in truth, I was driven to the 10 because my 6 had become unusable. And I'm now looking - I use CPU Dasher X, and I'm at about - I'm less than half of the phone's original clock rate on my 6, for no good reason that I can see, when it's plugged in, when it's charged at 100%. I believe they're looking at age rather than battery capacity and just assuming that, oh, well, if the phone's that old, it just must be hardly able to get off the ground now.

Well, no, not my phone. My phone's in great shape. But I don't get any credit for that. So, I mean, so I'm absolutely going to replace the battery. For 29 bucks it turns it into a new phone. Anyway, I'm really very upset. I've been talking about what a catastrophe iOS 11 has been with all of the crazy bugs. And it rendered my phone, my perfectly functional iPhone 6 completely unusable. And it's not like these things are cheap, as has often been noted. So anyway, I think I'm done ranting about my iPhone experience. I'm just, you know, wow.

**Leo:** I would love you to get one of those battery checkers.

**Steve:** I'll do it, for sure.

**Leo:** To just see what percentage, how many charges and what percentage is left.

**Steve:** And so does it pull data from the phone?

**Leo:** Apparently.

**Steve:** Do you let it run, and it runs the battery down to see how long it takes?

**Leo:** No, no, no, no, it doesn't have to do that. It can just - there's information offered by the chip, and somehow it gets that information.

**Steve:** Okay, cool. I will.

**Leo:** Yeah, there's a number of those. And I'd just be curious. I mean...

**Steve:** Yeah.

**Leo:** I would hope Apple's - I know they're not doing it by age because, if you put a new battery in, by all reports it fixes it.

**Steve:** Well, but, see, the battery is a subsystem itself. So it would know that it was a new battery. It would know when that battery was installed. It probably has a serial number and a battery management processor and, you know...

**Leo:** Right, I'm sure that's the case.

**Steve:** ...all kinds of crap.

**Leo:** Yeah.

**Steve:** I wanted to alert our listeners that we have another Humble Bundle, and this time it's Python books, which I think is probably right in the middle of many people's sweet spot. You know, Python is just really taking off. I'm encountering, as I have said over and over, I mean like more and more. And I had a very respected top-level programmer friend who has been a friend of mine for - how old am I now? - like 40-plus years, who programs in everything, tell me that he has never been more productive than he is in Python.

**Leo:** Yeah, it's a great language.

**Steve:** So, and it is my intention to rewrite the UI and the high-level portions of SpinRite 7 in Python. The idea is that the SpinRite 6.1, .2, .3 stuff, that will be sort of the testing bench for the new low-level drivers. And then I'm going to scrap it and keep all of that low-level work, but then write a whole new front-end for SpinRite, adding a whole bunch more features for SpinRite 7.

**Leo:** I would look at the reviews of these books before you buy them.

**Steve:** Yes.

**Leo:** These are not my favorite Python books, by any means.

**Steve:** And I'm glad you said that. I was going to say that. I don't know Packt, P-A-C-K-T.

**Leo:** They're all from Packt, aren't they.

**Steve:** Yes, they are a Packt book bundle. And so I think they may be along the cheesy

side. But they're also not very expensive.

**Leo:** Yeah, doesn't hurt to have it. I mean, I do the O'Reilly books pretty much. "Learning Python" is excellent, "Programming Python," those are classics.

**Steve:** Yup.

**Leo:** But there are some actually very good Python books out there.

**Steve:** Yeah. And you can't go wrong with O'Reilly. Tim is not going to be publishing junk.

**Leo:** Yeah, yeah.

**Steve:** And finally, in miscellany, we've had a lot of fun, Leo, over the years, laughing at robots falling down. There is a recently posted YouTube video that I linked here which is rather phenomenal. This is our friends at Boston Dynamics who are doing these amazing robots. This is Atlas, which is their human anthropomorphic robot, and it is doing some incredible things now, including a back flip. So this thing isn't just able to walk upright, bipedal, but it's able to do some incredible work. So if anyone is interested, I've got the link to it. And, yup, there it is. Look at this thing.

[YouTube clip]

**Leo:** And for those who aren't watching but listening, it's doing back flips, moving around...

**Steve:** Oh, Leo. I don't want to meet this thing on a...

**Leo:** Oh, I know. Look at that back flip.

**Steve:** It's, wow.

**Leo:** It's better than a gymnast.

**Steve:** It's incredible. But, I mean, when you think about what is necessary to do that, it's just stunning.

**Leo:** Yeah. Good balance, yeah.

**Steve:** So, wow.

**Leo:** And they have videos of it running out in the field, too. Those are the ones that scare me, you know, when they go outside.

**Steve:** Yeah.

**Leo:** And they start chasing people. That's what scares me.

**Steve:** That's right. Skynet is circling above.

**Leo:** Can't escape it.

**Steve:** So I got a note which prompts me to explain something from an Adriaan Stander, who's in Le Havre, France. This was dated two days before the new year, the 29th of December, was asking a question about SpinRite's levels. He said: "Dear Steve, I'm a longtime SpinRite owner, which I bought not because I had a problem with a drive, but as a reward for your work on Security Now!."

**Leo:** Nice.

**Steve:** Well, thank you, Adriaan. "I have never had to use it in anger" - or desperation, I suppose. He says: "...(i.e., for data recovery), but I do use it from time to time for disk maintenance." And of course I would argue that's why you've never had to use it in anger is that you're running it occasionally to keep your disks from ever having a problem, which actually has been proven as much as possible to prove such a thing, to work.

He says: "Can you perhaps sometime in the future" - well, technically this is, but only five days - "on a Security Now! podcast, explain the different levels of SpinRite in a bit more detail than what comes up when you run SpinRite? In particular, can you please give some practical examples of when levels 1, 3, and 5 may be useful? Kind regards, Adriaan Stander."

Okay. So for anyone who knows SpinRite 6, you're aware that the UI promotes the use of 2 and 4, levels 2 and 4. And that's the ones we talk about. I'm probably going to whittle those down even further. In the old days, SpinRite had a bunch of levels where I was using different levels of surface defect analysis. Level 1 has always been enforced read-only, just because I thought it should have that. That is, it will only, only, only ever read, and it will notify you if there's a problem. It will not ever write to the disk to fix it. So Level 2 is sort of the read-only level where it only reads unless there's a problem, and then it will examine that location to determine whether the problem is soft, and it could be fixed, or whether it is safe for data to remain there in the future.

I don't even remember now what Level 3 is. It's somewhere between 2 and 4. Literally, I can't remember what Level 3 is. I mean, it has some different features, but it's been a while since I was down in the code. Four goes beyond 2, which is read-only, and writes by default. So it does a full pattern test, well, as much of a pattern test as contemporary drives need. What's happened over time is that so much technology has been installed

into the drive's read and write channel that it absolutely no longer makes sense to imagine that user-provided data can result in knowable flux transitions on the drive. Before drives got as smart as they are, you actually could determine, you actually were able to write data that produced specific events of magnetic flux on the disk. And that was the big breakthrough in SpinRite 3.1 was it reverse-engineered the write channel in order to deliberately find defects.

Now there is so much going on between the data we write and what happens magnetically that there's no way for the user to control that. So what I've settled on, which is proven to be effective now because SpinRite 6 has been around long enough that we've acquired a lot of history, is it reads what's on the sector. It inverts it and writes it back, reads it to make sure that it was able to read the inverted data, then inverts it again, writes it, and reads it back. So essentially it flips all of the data bits into their non-one or zero state and then back again and makes sure that that process succeeds. Often in the process that shows the drive that there's something wrong with a sector it wasn't aware of, which causes it to then take action, map that data, map that sector out of use, and swap a spare sector in, which then is where SpinRite writes the final data. So that's part of that process.

And 5 just does that even further. It performs additional testing of the surface, but not testing that I'm convinced has ever been useful. I think I just - I was reluctant when I was working on 6 to prune back the options too much. And the user interface, by the way, does show you what Level 3 is. If you bring up the normal UI, you're able to step through 1, 2, 3, 4, and 5. And I describe there in the text what each of the levels does. And so it's possible to answer that question just by looking at the UI.

But basically 1 absolutely never writes anything, period. It can't. Which doesn't mean the drive won't still perform its own autonomous sector swapping, if you were to read a sector which scared the drive. And anyway, so it's very fast and may induce the drive to make some changes. Two definitely does. But it's read-only unless it finds a problem. So that's the one we recommend for use on solid-state drives where you don't want to read and write to them needlessly. And 4 is what we recommend for spinning drives, although because it's transferring so much data, remember, it's writing the entire drive, reading the entire drive, writing the entire drive, and reading the entire drive, which just takes time on big drives.

The first thing that will happen with SpinRite 6.1 is that time it takes will be dropping dramatically. As I have said in the past, I did clock it before, I clocked the prototype driver that I had written at half a terabyte per hour. So that means it'll be able to do a multi-terabyte disk in a reasonable amount of time. So that'll be good news for everybody who will be able to get that at no charge, as I have promised.

And one bit of closing-the-loop feedback from a Dallas Haselhorst, who said - oh, yeah. He said: "Steve, on Episode 642, in relation to SpinRite, you mentioned you should not use swap with SSD. I read about this quite extensively some time ago, and I was unable to come up with a solid answer one way or another, so I was surprised to hear you give it a definitive thumbs down. Does it matter which OS you are using, for example, Windows or FreeBSD? If the system has plenty of RAM, would adding swap cause any harm? Thanks."

Okay. So, yeah. I'm adamant about it. I don't know whether Windows is aware, but I sure hope it is because on a system that doesn't have an alternative place to write to put a swap file, I'm worried that the paging file is being written a lot. What we do see is a lot of page file activity on traditional Windows. And I do know from an experiment that a good friend of mind, Mark Thompson, performed that a Linux system will burn out a

compact disk where there's a swap file on it in very short order. So I don't - it's impossible to know whether OSes, or I should say I don't know whether OSes are smart enough to recognize that they are swapping on an SSD and to dramatically limit that. But the problem is at least Windows seems to swap when it has no good reason to do so, just because it can.

So one of the things that I have seen done is for a RAM disk to be established. And as far as I know, even on late-model Windows systems, you can still set up a RAM disk, although it's expensive because that RAM of course becomes unavailable to your system, but then set the swap drive there. That's something we used to do in the old days, a long time ago, in order to speed things up. But I'm just hoping that OSes are being faithful about this because it really does needlessly write to an SSD, even when really you could argue there's nothing, no need to swap when you've got a ton of RAM and you're not at max.

Okay. Finally, the strange story of "Extended Random." Or, as I titled this podcast, "The NSA's Fingerprints." Matthew Green, who is often quoted on this podcast, Johns Hopkins University cryptographer and now professor, made some interesting observations. He said there is additional circumstantial evidence to strongly, well, actually I'm saying, it's my voice, there's additional circumstantial evidence to strongly suggest that the United States NSA, the National Security Agency, was attempting to compromise encrypted communications.

As I mentioned at the top of the show, for reasons that were never clear, RSA Security, the very well-known inventors of public key cryptography and publishers of a very popular cryptography package known as BSAFE, were found years ago to have set as default an arguably insecure pseudorandom number generator, a so-called DRBG, Deterministic Random Bit Generator, using some elliptic curve functions whose magic numbers were never adequately explained. The magic numbers came sort of like from whole cloth. And it's like, okay, who chose those? And it was never really clear. But it looked like this came from the NSA via the NIST as part of the standards process.

But whenever you use magic numbers in something, moving forward especially, there has to be evidence provided of where that number came from, why that number was specifically chosen. We didn't have that with this elliptic curve DRBG. And moreover, because it uses elliptic curve, which is an expensive algorithm in terms of time, it was the slowest of the four. There was a symmetric cipher DRBG that would have been knowably secure and faster, which RSA didn't choose to use as their default. No explanation given. And as I mentioned, Reuters also apparently, well, Reuters did report that RSA was apparently paid $10 million from the NSA to put that into their BSAFE library. But it was sort of off in the, I mean, the whole RSA package wasn't clearly in use. No evidence of its deployment could be found. And that's what changed a couple weeks ago.

The trouble with the actual exploitation, or the compromise, of the Dual EC DRBG was that sufficient bits generated by it had to be obtained by an attacker in order to determine the internal state of the generator. So you need to see enough of its output in order to have enough leverage against it. And the normal TLS connection handshake, which as we've often discussed does contain some nonces, some pseudorandom data that has to not repeat, the amount of entropy in the handshake falls just shy of providing as much nonce information as an attacker would need to compromise the future numbers that the DRBG was going to be spitting out.

So now enter what's known as the Extended Random extension to TLS. As Matthew writes, he said: "The Extended Random extension is an IETF draft proposed by an NSA employee named Margaret Salter," he says, parens, "(at some point the head of NSA's

Information Assurance Directorate, which worked on 'defensive' crypto for the DoD), along with Eric Rescorla as a contractor." And I've seen Eric's name on a number of RFCs through the years, so he's well known in the community. So again, this Extended Random extension to TLS was proposed by an NSA employee.

So this Extended Random extension was never adopted. But it was a means of asking a TLS endpoint to provide more random data, more nonces, essentially. Matt wrote: "It's important to note that Extended Random by itself does not introduce any cryptographic vulnerabilities. All it does is increase the amount of random data, the nonces, used in a TLS protocol connection." He writes: "This shouldn't hurt TLS at all; and, besides, it was largely intended for U.S. government machines." He says: "The only thing that's interesting about Extended Random is what happens when that random data is generated using the Dual EC algorithm. Specifically," he writes, "this extra data acts as 'rocket fuel'" - his words - "significantly increasing the efficiency of exploiting the Dual EC backdoor to decrypt TLS connections."

Okay. So there's that. But all of this still remains academic, or did until two weeks ago because what happened was some researchers discovered that older Canon printers were unable to connect with TLS v1.3. Remember that pretty much the world is now at TLS v1.2. 1.3 is causing some problems because it's hostile to middleboxes. It's hostile, for example, to corporate enterprise proxies that want to be able to filter HTTPS connections and see into those. And it's been hostile to apparently large datacenters who want to have a similar proxy on their border and decrypt all traffic within the datacenter, which makes people feel a little bit queasy. So 1.3 has sort of hit some problems, some roadblocks as a consequence of the fact that its security is so good that it's hostile to some of the insecure things that people are currently doing with TLS.

Well, it turns out that the 1.3 specification was using an extension identifier, I think it was 4.0 if memory serves, which was also used by the proposed, but never adopted, IETF draft which the NSA proposed for Extended Random. In other words, this Extended Random extension was proposed. It was going to use a certain extension number to tack itself onto the existing TLS protocol. Didn't get adopted. Went nowhere. People forgot about it. And as a consequence of the fact that the extension that it was using was unused, it became used as part of the official TLS 1.3 work.

Then some Canon printers were found to be unable to use TLS 1.3. So some researchers thought, okay, what's going on here? Why can't these printers connect over 1.3? Why? Because those printers were actively using this never-deployed Extended Random extension and the Dual EC DRBG random number generator for their TLS. So no one was ever aware that the elliptic curve DRBG had actually been deployed, nor that this Extended Random Extension had ever been used. It turns out they were both in the RSA BSAFE library. And for whatever reason, and for whatever history, Canon chose to use all of that. And everything was fine until - that is to say, it went undetected until TLS 1.3 reused the extension of the never adopted proposed extension from the NSA.

So again, we're never going to know for sure what was going on here. But what we do know now that we did not know before is that the RSA BSAFE library added the Extended Random extension to provide more Dual EC DRBG bits in the TLS handshake, which the NSA asked for, the industry never adopted and ignored, but RSA went ahead anyway and stuck it in their library, which Canon printers then used. And now we finally got this last bit of information.

So none of this is proof. But as we know, this is the nature of these things, that there will likely never be absolute proof found. But one could argue that this is about as much proof as you could ever get that something was going on behind the scenes where the

NSA was hoping that what would become standard would end up being put into products where they could connect with them and compromise or watch the connections occur and then compromise the handshake nonces and then be able to encrypt the data. It doesn't look like that happened, but it does give us - it certainly looks like an NSA fingerprint in this whole process. Wow.

**Leo:** Very interesting. Busted.

**Steve:** Yup, exactly.

**Leo:** Ladies and gentlemen, we've come to the conclusion. Two liters of coffee and two hours of security has brought us to the end of this fine program. Thank you, Steve. We do this show every Tuesday, right after MacBreak Weekly, about 1:30 Pacific, 4:30 Eastern, 21:30 UTC. If you want to tune in and watch, you can at TWiT.tv/live. But we also have on-demand audio and video available. Steve's got it at GRC.com. You also get transcripts there, a few days after the show. Steve gets Elaine Farris to write everything down, and that way you can read along or search, which is really a nice feature.

GRC.com is Steve's home on the Internet, short for Gibson Research Corporation. While you're there, pick up a copy of SpinRite, the world's best hard drive recovery and maintenance utility, and check out all the other freebies Steve has there. You can leave him feedback there at GRC.com/feedback. But I think probably the easiest way is Twitter. He's @SGgrc on Twitter, and he takes DMs from anyone, of any length. We get a lot of questions for the show from there. You can also follow him, and it's a good way to kind of have a constant radar on security issues because you tweet all week, I know.

And then we have audio and video at our website, TWiT.tv/sn. Or you can subscribe, you know, if you use a podcast client on your phone or your tablet or your desktop, you can use any podcast client to search for Security Now! and subscribe in that way. You'll collect all the episodes, the best thing to do. Now, you'll be starting from now, but you can go back in time at TWiT.tv/sn and get the old shows, all 600 whatever it is, 30 - what did you say, 630 - 644.

**Steve:** Six hundred and forty-four.

**Leo:** Jiminy Christmas.

**Steve:** Woohoo.

**Leo:** Jiminy Christmas.

**Steve:** And going strong.

**Leo:** Let's do our weekly bitcoin check. Have you checked your bitcoins lately?

**Steve:** Well, they did take a hit over the holidays but are beginning to creep back up. I saw it at like 14,000 when I looked this morning. So I'm just kind of keeping an eye on it.

**Leo:** Not worrying about the bitcoins in the corner, the 50 bitcoins in the corner. I still can't find my password, so...

**Steve:** Oh, in fact I heard - I got a nice note from a listener of ours who, back when we discussed bitcoins...

**Leo:** By the way, that was our holiday episode; right?

**Steve:** Yes. And in fact it was...

**Leo:** I know what you're going to say because I got that email, too.

**Steve:** Oh, okay, yeah.

**Leo:** Because I created the wallet during that show.

**Steve:** Right. Well, remember that we pointed people at the Bitcoin Faucet, which at the time allowed - it was like a little drip. It would give you a little drip of bitcoins. What that drip was out of that faucet is now worth $743.

**Leo:** Each drip.

**Steve:** Each drip. And one of our listeners thanked us for $743 because he cashed in his drip from his wallet that he still had and made some money off of it. So, yeah.

**Leo:** I'm glad you cashed in your drip. We should all cash in our drips.

**Steve:** Which is not a phrase I expected to be saying.

**Leo:** But we all learned something in that bitcoin episode. Thank you for suggesting that as a holiday.

**Steve:** I got a lot of nice feedback from people who were happy to see that again. So

that was a good thing. And Leo, speaking of good things, I will be back next week with you.

**Leo:** Au revoir, Steve.

**Steve:** And we will catch everybody up on what has happened since today. Bye.