VPN-geddon Denied

Description: This week we look at Microsoft's force-feeding of Windows 10 feature updates, the creation of a tool to keep Win7 and 8 updates freely flowing for free, the continuing evolution of a new highly secure programming language, an update to Microsoft's RDP client for iOS, Avast and AVG in the doghouse, some VERY severe authentication bypasses in OpenBSD, and a note about the WireGuard VPN. Then we take a look at the report which every security website breathlessly covered - and got wrong.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-744.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-744-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve is here to talk about forced Microsoft Windows updates. We're going to talk about Microsoft turning to Rust for systems programming, a brand new VPN Steve likes an awful lot, and why you might have read some headlines about VPN-geddon you don't have to worry about. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 744, recorded Tuesday, December 10th, 2019: VPN-geddon Denied.

It's time for Security Now!, the show where we talk about your security and privacy online with our good friend, Steve Gibson. Hello, Steverino.

Steve Gibson: Yo, Leo. Great to be with you again.

Leo: Good to see you.

Steve: There was a piece of news that not a single tech press publication failed to cover.

Leo: Incorrectly, no doubt, yeah.

Steve: Incorrectly.

Leo: Uh-oh. Okay.

Steve: Involving what was claimed to be a horrible, critical, end-of-the-world, VPN-agnostic problem in Linux and all of the *Nixes and the *BSDs and iOS and Android, I mean, it just covered the territory. So I titled our episode "VPN-geddon Denied" because this is not VPN-geddon. But we'll have fun talking about it. It's interesting from a technology standpoint. I don't mean to get on the case of the tech pubs. They're not developers. But it's such an interesting thing that happened that I think it would make a great topic for today.

But we're going to take a look at Microsoft's beginning to force-feed Windows 10 feature updates onto people who, well, after more than year presumably don't want them, but they're going to get them anyway. We have the creation, and you could have seen this coming, of a new tool to keep Windows 7 and 8 updates freely flowing for free, after Microsoft attempts to cut them off after January.

We have the continuing evolution of a new, highly secure programming language. An update to Microsoft's RDP client for iOS, of all things. Avast and AVG are in the doghouse. We have, now, in this case, some actually truly severe authentication bypasses in OpenBSD such that anybody listening to this, if you have OpenBSD exposed to the Internet, you need to make sure you've updated it recently. We also have a note about a VPN known as WireGuard that I ran across earlier this summer. And it's been on my radar. I'm planning to do a deep dive for our listeners into it. But it popped up actually on my radar for today. So we'll talk about that. And there was sort of an interesting coincidence that involves it when I was in Sweden that I'll briefly talk about.

And then we're going to take a look at this report of, well, every security website breathlessly covered and got wrong. And speaking of insecure programming languages, we're talking about one that is a language written for security, with security foremost, is evolving. But our Picture of the Week is such a perfect example of just how anti-secure a language, in this case PHP, can be. So I think another great podcast for our listeners.

Leo: Sounds very exciting. Very exciting. Steve?

Steve: So Leo, our Picture of the Week is just such a perfect example of how programmers can so easily get themselves in trouble and get tripped up when a language they're using really wasn't designed for the application it's being used for. In this instance, for those who don't have video, this is an example of an equation testing for the equality of two hashes.

And so there's a string, 240610708, which is hashed with MD5. And the fact that it's MD5, that's not significant here. We know that that's been deprecated, but that's not the point. And then there's an equality sign joining it to another hash of a completely different string, QNKCDZO. And we know that these two different strings should, with an extremely high probability, hash to different results. So they should not be seen as equal.

But automatic languages, languages which jump in and do things on behalf of the programmer to make the programmer's job easier, often do something known as "type mangling," where it will see that you're trying to compare two things, and it'll go, oh, okay. Well, MD5 produces a string of its output, so we're comparing two strings. Then, unfortunately, PHP looks at the strings, and it sees something that it thinks is a number. The kicker here is that both of these hashes begin with the characters 0e. And that looks like, to PHP, it looks like scientific notation, that is, 0e followed by an exponent.

Well, anything, zero raised to any power, any exponent of zero is going to be zero. So this equality sees that you're comparing zero versus zero, even though the actual hashes and the strings are completely different. They just happen to collide with the first two

characters, which as a consequence of PHP's overly helpful type conversion, or mangling, says, oh, look, I don't even have to do the math here. "0e," that means anything that follows it is going to not matter.

Leo: I would blame the programmer, not the language in this case because you just didn't use the right type for these hashes, that's all.

Steve: Correct.

Leo: It's just a string converted to a number. You don't have to - and that's the programmer's fault, honestly. Admittedly, PHP's defaults are probably not security forward. But still.

Steve: Well, and that's the point is the defaults are to be helpful and to say, oh, look how easy this is to use. You can learn this in an afternoon. And, yeah, but don't write...

Leo: Yeah. You might be tempted to say, oh, just going to cast a string to a number, and it'll be fine. But any, I mean, I think a decent programmer is going to know better, I would hope.

Steve: Well, and the problem is that these languages allow people who don't really have the chops to write code that they should not be writing.

Leo: Yes. Like security code.

Steve: Like, oh, I could write some. Yeah, I'll check the hashes to make sure they're the same. I mean, you can see this. You can imagine this happening on some web page that someone wrote in PHP. And it's like, oh, look. Looks great to me. It's like, yeah. Except here's a little glitch. Anyway, this is sort of subtle, but I thought it was a perfect example of a way that a language that is trying to help you can actually hurt you. And the fact is we see these sorts of things all the time. You really need to understand what your language is doing for you in order to write code that always behaves the way you expect it to.

Leo: Yes. And I think this is a case where a better designed language would make it harder for you to make that mistake.

Steve: Yes.

Leo: That's why we like statically typed languages, for one thing; right?

Steve: Yes. Or strongly typed. I'm a real fan...

Leo: Strongly typed, yes.

Steve: ...of strongly typed language because there, if you're declaring what everything is, then the compiler can help you to find your own mistakes. You're trying to assign a string to an integer, and it's like, whoa. Whereas PHP says, yeah. We know how.

Leo: Yeah. I'll be glad to do that.

Steve: I'll convert that for you. And it's like, whoa. So anyway, I just...

Leo: Another argument for not writing your own security code, but using libraries that are well designed, too; right? I mean, a good library's not likely to do that.

Steve: Yes, exactly. Good, well-designed libraries written by people who have an expertise in that thing can then produce for you an API which encapsulates that security knowledge and language-specific knowledge so that that work is being done for you. And so, for example, if there was an API that was to compare two hash buffers to make sure that they're the same, it would have done the right thing if it were written by somebody who should write such a library.

Leo: Right.

Steve: So Microsoft has started forcing feature updates on people who apparently don't want them. Windows 10 version 1809 - not to be confused with 1909. 1809, as we know was first, well, ostensibly made available more than a year ago on November 13th of 2018. That was the one that just kept having so many problems that they ended up not releasing it until several months into this year, 2019. But that release for Windows Home, Pro, Pro Education, and Pro for Workstations, its end-of-service date is next May 12th of 2020. So it's still a ways away.

But after May 12th of 2020, any systems still running what was the October 2018 update will no longer receive any future quality and security updates. Therefore, in a deliberately planned preemptive move, since after all it has been more than a year since its original intended release, which was November 13th of 2018, Microsoft last Thursday announced that they were going to start pushing people forward.

Last Thursday they said: "Beginning today, we will slowly start the phased process of automatically initiating a feature update for devices running the October 2018 Update Home and Pro editions, keeping those devices supported and receiving the monthly updates that are critical to device security and ecosystem health." They said: "We're starting this rollout process several months in advance [five or six] of the end-of-service date to provide adequate time for a smooth update process."

And you know, you and I, Leo, we kind of go around about this. We're in agreement. I mean, we understand Microsoft's need to drag their legacy stuff kicking and screaming forward, and it makes sense. They're wanting to have pretty much a single build that they are then continually rolling forward. And we talked about it at the time, when they would allow an individual under the advanced update settings of the versions of Windows that allow that to go in and forestall a feature update, not the security update, but the feature update by up to 12 months. And so basically people have been able to push this off for that length of time.

Leo: But time's up, Charlie.

Steve: Yeah. And given all the trouble that that particular one had, anybody would be understood, especially if they don't want any feature updates, if they're like, yeah, I'm happy with what I have. Leave me alone. Well, you've got a year. Now, as you said, time's up. So Windows will no longer accept the "Thanks anyway, but not yet" response to whether they want to receive a feature update. And again, basically it looks like six months ahead Microsoft is saying, okay, you've had more than a year. We're not really lowering the boom until May of 2020, but we're just going to start moving things forward.

So anyway, if anybody sort of has a machine that's been stable and settled down, and then suddenly they get one of those, hold on a second, we're improving your Windows experience or updating, and you get the little rollercoaster balls for, like, way longer than usual, it's probably because you fell into this, okay, no more waiting.

Leo: Yeah. And I think if you're an enterprise license person, you have much more flexibility on that. It's just the regular users that have to do the upgrade.

Steve: Well, my favorite is the Long Term Servicing Channel. You're completely released from all of this if you're on the LTSC bandwagon. If you're an enterprise user, and you're like, no, we really do want to stay with what we have, then Microsoft says, yeah, okay, fine.

And in a bit of "we should definitely have seen this coming" news, a group of Windows enthusiasts who hang out over in the My Digital Life forum have come up with something they call "BypassESU" utility, where ESU stands for Extended Security Updates. I have the link, if anyone is curious, in the show notes. So two months before it'll even be needed, we already have a hack to trick Windows 7 and 8 into continuing to receive another three years of security updates after the otherwise final January 14th Patch Tuesday, next month.

So the story goes, last month Microsoft released a test Windows 7 ESU update that was KB4528069. It allowed IT administrators to start the verification process that their systems would be compatible with the upcoming ESU, the Extended Security Updates process. And I have also a link in the show notes to that update.

And so there are a couple interesting points in here. In their summary describing the update, they said: "This optional non-security update will help you verify that your eligible Windows 7 SP1 and Server 2008 R2 SP1 devices can continue to get Extended Security Updates after the end-of-support date of January 14th, 2020." So let's see. They said no security content. "This update is a test package we recommend that you deploy in your test environment. Install this update on your on-premise devices that are eligible for Extended Security Updates."

This update can be installed on x64-based architecture for Windows Server 2008 R2 and also either x86 and x64-based for Windows 7 SP1. They said: "This update is not applicable for Windows 7 Virtual Desktop (WVD) and Windows 7 Embedded. Installing this update has no impact on getting security updates between now and January 14." So it's only for continuation. Under installation instructions, install the update from one of the release channels such as Microsoft Update Catalog, meaning that anybody can do that.

And then they said for prerequisites: "You must have the following installed on your on-premise device before you apply this update." They said: "Install the following SHA-2 code-signing support and servicing stack update." And then there's a link to that. Of course we've talked about that. That's where some months ago Microsoft stopped dual signing the updates, I think it was this summer, so that the newer updates are only being signed with SHA-2, not also SHA-1. So obviously you would need that in order to, I mean, and everybody probably already has that. And that creates the SHA-2 code-signing support. You're supposed to have the servicing stack update, and there's another knowledge base article from March 12th, and be up to date with the monthly rollup. And then the last one is install and activate the ESU key. For information about how to install and activate the ESU key, see "How to get Extended Security Updates for eligible Windows devices" blog on the Microsoft Tech Community website.

So what happened was some guys went in, and they figured out basically how to short-circuit this ESU key requirement. So just to be clear, doing this violates licenses. It's technically against the law. So that hasn't kept people, for example, for decades of figuring out how to create hacked Windows OS installation key checks. That's a hack that's been around forever. And also this is the kind of thing that Microsoft will likely respond to.

So I wouldn't expect this particular bypass ESU tool crack, is what essentially it is, to work forever. I expect that, I mean, and it'll be interesting to see with how much fervor Microsoft pushes back on this; whether they're going to say, wait a minute, we don't want hackers to be continuing to get Windows 7/Windows 8 updates. Basically, if you're not an enterprise who's paying for them, you need to do that. So anyway, we'll keep our eye on this. Again, I just sort of chuckled when I saw, okay, well, yeah, it figures that some hobbyists would figure out how to do this. And so we'll see if this is going to be a cat-and-mouse game. Microsoft will patch it, somebody will come up with a way around it again, blah blah blah.

And I did note sort of the irony of the fact that Microsoft, if they were to defeat this ESU bypass, would have to do it with an update, which of course is what the ESU bypass allows you to receive. So if you've got the ESU bypass installed, then you're able to get the update from Microsoft which probably breaks it and causes you to no longer be able to get any more updates. So maybe the thing to do is just to say, okay, fine, I'll upgrade to Windows 10. We talked about how that can still be done for free last week.

As I've been talking about recently, on recent podcasts, we're at the point now where - and Leo, I had so much fun listening to your Mac Pro coverage.

Leo: Isn't that wild?

Steve: I mean, the size of the RAM.

Leo: My god.

Steve: Was it $69,000? Did I hear that right?

Leo: I think we maxed it out, I think at $61,000.

Steve: For a single machine.

Leo: That might have included the $6,000 monitor. I don't remember. It's over $50,000, yeah.

Steve: Wow. Anyway, so talk about having...

Leo: But honestly, I mean, going back in time, I remember buying a DEC PC for $10,000.

Steve: I think my first IBM XT was 10-plus. I wanted both monitors. I needed the 10MB hard drive.

Leo: Exactly.

Steve: I think the hard drive was $5,000.

Leo: Exactly.

Steve: It was 5K to not have to use floppies.

Leo: So we're talking a lot of money, but we're also talking a lot of computer. I mean, it's a supercomputer. It really is.

Steve: Well, yes, it is. Which sort of brings us to the point that I've been talking about recently is that we have, I mean, there's like a 64-core Threadripper is now the term that's being used?

Leo: Yeah, that's next for AMD's Ryzen.

Steve: So the history of programming languages has been one of trying to ring as much performance as possible out of the system.

Leo: Not anymore.

Steve: And there certainly are places where you're doing high-end 3D modeling or massive matrix work, or you're modeling atmospheric perturbations, I mean, that are like incredibly number-crunchy. But many of us are sitting in front of a word processor which we would like to not have hackable if we make the mistake of opening a JPEG in the word processor. That doesn't take a lot of processing power, to get spellcheck that works with reasonable performance, given the engine which is powering our machines these days.

So as I've been talking about, it is really making sense for us to sort of stand back and say, okay, the problem we have today is no longer that we're not able to type into our

word processor at a speed which it can keep up with. The problem we have today is that all of our crap keeps getting hacked. And so let's fix that. Let's fix these buffer overruns. And, I mean, like, really, let's come up with a language which no longer allows those mistakes to be made, a language where you don't give the programmer - and this is where the programmers chafe a little bit - you don't give the programmer the ability to get themselves in trouble. And that PHP example is a perfect instance of an anti-secure language.

So Microsofties are talking more and more about something called Project Verona. It's the codename for their increasingly customized, tightened, and hardening language which is reminiscent of and has strongly apparently been influenced by the Rust programming language. We've talked about Rust a little bit. There was an interesting blog post last week by Adam Burch, actually it was a little older than that, but he's a software engineer in the Hyper-V team. And he shares his experience. The tags on the post are memory safety, Rust, safe systems programming languages, and secure development.

He wrote: "This Saturday, 9th of November, there will be a keynote from Microsoft engineers Ryan Levick and Sebastian Fernandez at RustFest Barcelona. They'll be talking about why Microsoft is exploring Rust adoption, some of the challenges we've faced in this process, and the future of Rust adoption at Microsoft. If you want to talk with some of the people working on how Microsoft is evolving its code practices for better security, be sure to attend the keynote and talk to Ryan and Sebastian afterwards.

"This blog describes part of the story of Rust adoption at Microsoft. Recently," he wrote, "I've been tasked with an experimental rewrite of a low-level system component of the Windows codebase," and he said, "(sorry, we can't say which one yet)." He said: "Instead of rewriting the code in C++, I was asked to use Rust, a memory-safe alternative. Though the project is not yet finished, I can say that my experience with Rust has been generally positive. It's a good choice for those looking to avoid common mistakes that often lead to security vulnerabilities in C++ code bases."

And he said, under the heading "Great Dev Experience," he said: "For C++ developers used to writing complex systems, using Rust as a developer is a breath of fresh air. The memory and data safety guarantees made by the compiler" - and this is the key, by the compiler, not the programmer - "give the developer much greater confidence that compiling code will be correct beyond memory safety vulnerabilities. Less time is spent debugging trivial issues or frustrating race conditions. The compiler warning and error messages are extremely well written, allowing novice Rust programmers to quickly identify and resolve issues in their code. Visual Studio Code already has a helpful extension (RLS) which provides IntelliSense suggestions and syntax highlighting. Additionally, the Cargo build tool offers very helpful features around testing, documentation generation, and auto formatting."

As for the learning curve, he writes: "Thanks to a plethora of online documentation and very helpful compiler error messages, Rust has a pretty easy learning curve for someone like me who has used C++ for the majority of my career. There are tutorials aimed specifically at C and C++ systems engineers. In his talk at RustConf 2019, Jeremy Fitzhardinge at Facebook noted that he saw experienced C and C++ devs become comfortable with Rust in around four weeks and pretty fluent in eight. This aligns with my own experience.

"I participated in the annual Microsoft internal One Week Hackathon with one experienced Rust developer and one complete novice. Within three days, the novice Rust developer had written more than 1,000 lines of idiomatic Rust code. In addition to the great documentation, there are helpful tools like Clippy" - I don't want to know what that is - "which allow experienced C++ developers to jump right into coding Rust without" -

oh, is it going to jump out and ask, ooh, looks like you're trying to sort a buffer? Anyway, "...jump right into coding Rust without much direct assistance from those experienced with Rust."

He said: "As we expand the use of Rust inside Microsoft, I believe it will be prudent to start a Rust Reviewers group for any PRs that include Rust code. This will help novices in diverse teams get feedback from Rust experts, regardless of the specific problem domain. In general, new components or existing components with clean interfaces will be the easiest to port to Rust. The component I've been rewriting has been challenging, as there are many abstractions leaked from one layer to the next, requiring some preliminary refactoring before progress could be made." Which is to say there were approaches and idioms from whatever it was written in before, probably C, that don't map one-to-one onto Rust because of the different approach that Rust takes. So it required some reconceptualization of what the C code was doing in order to express it the way Rust wants things expressed.

But he's finishing, he says: "Keeping It Safe. To obtain the desired safety guarantees from Rust, strict guidelines must be placed around the use of the unsafe keyword." And I don't know Rust, so it's hard to say...

Leo: It's very C-like, yeah.

Steve: Yeah.

Leo: Unsafe says this is - yeah. Well, you can guess what that is.

Steve: Yeah. Well, what it must do is it must be explicitly turning off some of the things that the compiler would do in order for you to do something which you're acknowledging is unsafe. And so the compiler says, okay, well, I was about to slap you, but as long as you use that responsibly. Anyway, so he says: "Any calls to a Foreign Function Interface should occur in a wrapper function that provides a safe abstraction around it." So in other words, you're sequestering the dirty bits that you have to have in order to talk to other parts of the system that haven't yet had interfaces defined that were safe.

Anyway, I had a little bit more here, but all of our listeners have the gist of it. He ended up being very bullish. And he noted somewhere, I didn't want to skip this, he noted going back to C++, he said: "In general, using Rust has been a really great experience. I look forward to seeing more developers at Microsoft working on the language and working with the wider community on making the language an even better fit for some of the things we do here at Microsoft." So Microsoft is actively giving back to the Rust community. There are things that he talks about where, because of the size of the projects they're doing, they need more enterprise-grade source and module management than Rust has in a way that is useful for them. So they're actively working in an open sense to evolve Rust to be capable of being used within their own environments.

Leo: It's kind of interesting. They have, of course, TypeScript and other kind of safer functional languages they've developed, including, what is it, F++? F? So it's interesting they're using this open source solution. Rust is well loved as a systems language. And I think a lot of C programmers should be very comfortable with it.

Steve: Yes. Well, yes. And it is turning out, and this was some of the feedback from Facebook, it is holding its own in performance.

Leo: Oh, yeah. Rust is amazing.

Steve: So you're not taking a big...

Leo: No.

Steve: Yeah, it's not like dropping into a scripting language. I mean, you're getting the same level of performance. I found the line that I thought was really interesting in here. He said: "After writing Rust code, I find writing C++ much more frustrating..."

Leo: Oh, yeah.

Steve: "...since I can't rely on the compiler to ensure memory safety." So anyway, I wanted to sort of just put another pin in this notion that it's time now. As I said, lord knows our email clients are fast enough. So let's write them in Rust so they're not sources of ridiculous security flaws as they continue to be.

Leo: Nice.

Steve: Or some other truly safe language. So Microsoft's RDP client, which...

Leo: Beloved RDP client.

Steve: Yes, beloved RDP client for iOS is back.

Leo: Oh.

Steve: I don't think I was aware that Microsoft had a remote desktop protocol client for iOS. That might be sort of cool, though. So their RDP client for iOS had not been updated for more than a year. Then a couple of weeks ago they updated the client to version 10, put it up on the Apple App Store, then almost immediately pulled it down without saying anything. It just disappeared. And I saw that news, and I thought, well, okay. That's interesting. I didn't know there was an RDP client for iOS. And how would that work, exactly?

But based upon some hearsay from users, the short-lived version 10.0 Remote Desktop client would wipe all of the user's saved Remote Desktop settings when it was installed, which apparently is undesired behavior. One user posted: " It's because it wipes all your saved config inside the app."

Leo: Handy.

Steve: "It resets it to new." He wrote: "I managed to update before they pulled it." And then another user also encountered a similar issue. So there have been a couple people who concur. But apparently after force-closing the app and restarting it, the existing profiles were still present.

Anyway, last Wednesday the iOS RDP client reappeared as version 10.0.1 with that problem solved. And I have a link in the show notes to the App Store entry for it. And I was curious because, well, combining it and a VPN would be an interesting solution for remotely managing Windows systems.

So they say: "Microsoft Remote Desktop. Get work done from anywhere. By Microsoft Corporation. Use Microsoft Remote Desktop for iOS to connect to a remote PC or virtual apps and desktops made available by your admin. With Microsoft Remote Desktop, you can be productive no matter where" - you know. We all know what the features are going to be.

One thing that caught my eye, though, was that it said, under What's New for 3rd of December - so maybe that was the first release. Anyway, it's the version 10, which now is 10.0.1, does now have support for the Windows Virtual Desktop service. There's a brand new connection center UI, brand new in-session UI for switching between connected PCs and apps, a new layout for an auxiliary onscreen keyboard, and then this is - oh, improved external keyboard support. And then support for SwiftPoint Bluetooth mice. And I said, what? Mice?

Support for microphone redirection. Support for local storage redirection. Support for camera redirection. Support for new iPhone and iPad devices. Dark and light theme support. Control whether your phone can lock when connected to a remote PC or app. Collapse the in-session connection bar with a long press on the remote desktop logo. Then they finished, I guess feeling a little sheepish that they hadn't been doing anything for a year: "We're committed to making this app the best Remote Desktop client for iOS, and value your feedback. If you encounter any errors, you can contact us via Settings/Report an Issue."

So I just sort of wanted to put this on everybody's radar. I'm going to have to make some time to give this a try and play with it. I didn't realize that one of the features in iPad OS v13, they sort of snuck in mouse support. You get to it, and I found it, by going into the Control Panel. Then under Assistive Support you turn that on. And then a ways down is Devices. And you open that, and there's mice in there. So either a wired or a Bluetooth mouse can be used with your iPad, which I think would be kind of cool. I mean, I love my iPads. I've been loving them ever since version one.

Leo: Well, also the big screen would be nice for remote access; right? I mean, especially if you use iPad Pro.

Steve: Yes, iPad Pro, big screen. I mean, maybe you can do it just by touch. But anyway, I thought that was really cool. So just wanted to let people know that exists. I mean, if you ever have, I mean, you certainly - we know we don't want to have RDP exposed to the public Internet. There's just no way to do that. So you'd want to combine it with a VPN. But given that you VPN for security, for secure access to the RDP server, being able to do something like emergency reboot or something with your phone, that would be cool.

Okay. Enough Microsoft stuff. Avast and AVG are in the doghouse.

Leo: [Growling]

Steve: Uh-huh. Yeah. The Avast Online Security, Avast SafePrice, AVG Online Security, and AVG SafePrice extensions have all been pulled from the Mozilla add-on repository after they were found to be silently collecting far more user-identifiable data than was required to do their jobs. This story begins with a German web researcher and developer by the name of Wladimir - it's "W" rather than "V," so it's not Vladimir, it's Wladimir, I guess - Palant. And so we might first ask, does Wladimir have any street cred? I'd say so.

The first two paragraphs of his bio read: "My name is Wladimir Palant, and developing software is both my day job and a hobby. I became fascinated," he writes, "by Firefox extension development in 2003, and my Adblock Plus extension became so popular that I eventually co-founded Eyeo GmbH to continue its development." So this is the Adblock Plus, the guy who originated Adblock Plus.

He says: "By now Adblock Plus isn't available just for Firefox, but for Chrome, Opera, Safari, Android, and Internet Explorer, as well." He says: "My other notable Firefox extensions are Google/Yandex search link fix, as well as developer tools JavaScript Deobfuscator and Extension Auto-Installer." And he goes on talking about he has an interest in web application security and so on. So this guy knows what he's talking about. He was poking around the Avast and AVG extensions and was shocked by what he found. His announcement and takedown of these guys is titled "Avast Online Security and Avast Secure Browser Are Spying on You." I've got the link in the show notes for anyone who wants more detail.

He asks, rhetorically: "Are you one of the allegedly 400 million users of Avast AV products? Then I have bad news for you: You are likely being spied upon. The culprit is the Avast Online Security extension that these products urge you to install in your browser for maximum protection. But even if you didn't install Avast Online Security yourself, it doesn't mean that you aren't affected. This isn't obvious, but the Avast Secure Browser has Avast Online Security installed by default. It is hidden from the extension listing and cannot be uninstalled by regular means. Its functionality is apparently considered an integral part of the browser. Avast products promote this browser heavily, and it will be used automatically in banking mode. Given that Avast bought AVG a few years ago, there's also a mostly identical AVG secure browser with the built-in AVG Online Security Extension."

So his summary of findings. He says: "When Avast Online Security extension is active, it will request information about your visited websites from an Avast server. In the process, it will transmit data that allows reconstructing your entire web browsing history and much of your browsing behavior. The amount of data being sent goes far beyond what's necessary for the extension to function, especially if you compare to competing solutions such as Google's Safe Browsing. Avast's Privacy Policy covers this functionality and claims that it is necessary to provide the service. Storing the data is considered unproblematic due to anonymization" - he says, "(I disagree)" - "and Avast doesn't make any statements explaining how long it holds onto it." And actually some additional research that he did later explains probably why they're holding onto it.

So he says: "What's happening exactly? Using browsers' developer tools, you can look at an extension's network traffic. If you do it with Avast Online Security, you'll see a request to" - and it's a URL, uib.ff.avast.com, and that "ff" is probably Firefox, dot avast.com/v5/urlinfo. He says, "...whenever a new page loads in a tab." And he shows an example here of a screenshot, I have it in the show notes for anyone who's interested, of a snapshot of the data which is being sent.

He said: "So the extension sends some binary data, and in return gets information on whether the page is malicious or not." That is, it sends this data out, pinging the Avast server for is this page safe to allow the user to visit or not. The response is then translated into the extension icon to be displayed for the page. He says: "You can clearly see the full address of the page in the binary data, including query part and anchor. The rest of the data is somewhat harder to interpret. I'll get to it soon. This request isn't merely sent when you navigate to a page. It also happens whenever you switch tabs. And there's an additional request if you are on a search page." Get this. "This one will send every single link found on this page, be it a search result or an internal link of the search engine."

Leo: Well, that's not nice. I understand them checking the page before you log in. I mean, that's, by the way, why I don't like antiviruses. I've never recommended AVG or Avast for this reason. It was free. It's free for a reason.

Steve: Yes. I was going to say that. And I encounter it installed on, I mean, I don't want to say "just about every."

Leo: All the time, yeah.

Steve: But it's all over the place.

Leo: Now, they claim that they're anonymizing this, and it's not - but still.

Steve: Well, and he addresses the fact that studies have shown what degree of deanonymizing is possible, given enough data. And, okay. So he says: "The binary UrlInfoRequest data structure used here can be seen in the extension source code. It is rather extensive, however, with a number of fields being nested types. Also, some fields appear to be unused, and the purpose of others isn't obvious. Finally, there are 'custom values' there, as well, which are a completely arbitrary key/value collection." He says: "That's why I decided to stop the extension in the debugger and have a look at the data before it's turned into binary. If you want to do it yourself, you need to find this.message call in scripts/background," blah blah blah. And he explains how to do it.

So what's there? He says the interesting fields were the URI, the full address of the page you're on; the title, the page title, if available; the referer; the windowNum or tabNum, which is the identifier of the window and tab that the page loaded into. Then there's an initiating user action window event, how exactly you got to the page, for example, by entering the address directly, by using a bookmark, or by clicking a link. Then, whether you visited this page before, a "visited" field. The locale, your country code. A user ID, a unique user identifier generated by the extension, so that's going to be globally static for you for the life of your use of this. Again, completely unnecessary for them to have that. Except for the way they're monetizing, which we'll get to in a second.

They say: "For some reason this one wasn't set for me when Avast AV was installed," although he has that shown in his notes. There's a plugin GUID, a globally unique ID. Seems to be another unique user identifier, the one starting with "ceda" in the screenshot above. "Also not set for me when Avast AV was installed." A browser type and browser version; an OS and OS version. And he says: "That's merely the fields which were set. The data structure also contains fields for your IP address and a hardware

identifier, but in my tests these stayed unused. It also seems that, for paying Avast customers, the identifier of the Avast account would be transmitted, as well."

So he says: "What does this data tell about you? The data collected here goes far beyond merely exposing the sites you visit and your search history. Tracking tab and window identifiers, as well as your actions, allows Avast to create a nearly precise reconstruction of your browsing behavior: how many tabs do you have open, what websites do you visit and when, how much time do you spend reading/watching the contents, what do you click there, and when do you switch to another tab. All that is connected to a number of attributes allowing Avast to recognize you reliably, even a unique user identifier.

"If you now think, 'But they still don't know who I am,' think again. Even assuming that none of the website addresses you visited expose your identity directly, you likely have a social media account. There have been a number of publications showing that, given a browsing history, the corresponding social media account can be identified in most cases."

So, finally, "Isn't this necessary for an extension to do its job? No. The data collection is definitely unnecessary to this extent. You can see this by looking at how Google Safe Browsing works, the current approach being largely unchanged compared to how it was integrated in Firefox 2.0 back in 2006. Rather than asking a web server for each and every website, Safe Browsing downloads lists regularly so that malicious websites can be recognized locally." And then he cites a little blurb from Firefox: "No information about you or the sites you visit is communicated during list updates. Before blocking the site, Firefox will request a double-check to ensure that the reported site has not been removed from the list since your last update. This request does not include the address of the visited site. It only contains partial information derived from the address."

So Firefox demonstrates, and Google has demonstrated, it's entirely possible to offer this kind of service without, well, first of all, with local caching of known problematic websites, and it's merely a confirmation to double check. So he writes: "I've seen a bunch of similar extensions by antivirus vendors, and so far all of them provided this functionality" - that is, similar to Firefox - "by asking the AV app itself," not an online server. "Presumably, the antivirus has all the required data locally and doesn't need to consult the web service every time."

Then he later made an edit. He said: "I got a hint that Avast acquired Jumpshot a bunch of years ago. And if you take a look at the Jumpshot website, they list 'Incredibly detailed clickstream data from 100 million global online shoppers and 20 million global app users. Analyze it however you want. Track what users searched for, how they interacted with a particular brand or product, and what they bought. Look into any category, country, or domain." And on and on. He says: "So now you have a pretty good guess as to where your data is going." And again, Leo, I agree with you. This is free. You get what you pay for. You and I would argue that Windows Defender is now doing as good a job as this.

Leo: It's always done better than AVG and Avast.

Steve: Yes, exactly.

Leo: I mean, yeah.

Steve: Exactly, exactly.

Leo: The real irony of this is that these are companies that in theory make antimalware tools that one could argue are creating, by putting it on your machine, you're installing spyware.

Steve: Yes. I mean, this is extensive, I mean, admittedly it's a GUID. If you're a paying customer, they may know who you are. And I'm not suggesting that anybody who is taking advantage of this Jumpshot data - but, boy, Leo, if you go over to Jumpshot, it's chilling.

Leo: Oh, boy.

Steve: It's one of those deep research, learn who's searching for what. And so basically AVG and Avast users are being monetized, without their knowledge or permission, by feeding all of their browser activity, tab switching, and how they got to the page, and what they clicked on once they were there, their entire clickstream is being fed back into a database for monetization.

Leo: Yeah, I mean, just the fact that every link on every page I visited is also sent back.

Steve: Yes.

Leo: There's no security reason for that, so obviously they're gleaning, intentionally gleaning information from the pages I'm on. That's not okay.

Steve: Yeah. He concludes: "Avast Online Security collecting personal data of their users is not an oversight and not necessary for the extension functionality either. The extension attempts to collect as much context data as possible, and it does so on purpose. The Avast privacy policy shows that Avast is aware of the privacy implications here. However, they do not provide any clear retention policy for this data. They rather appear to hold onto the data forever, feeling that they can do anything with it as long as the data is anonymized. The fact that browsing data can usually be deanonymized doesn't instill much confidence, however.

"This is rather ironic, given that all modern browsers have phishing and malware protection built in that does essentially the same thing, but with a much smaller privacy impact. In principle, Avast Secure Browser has this feature as well, it being Chromium based. However, all Google services have been disabled and removed from the settings page. The browser won't let you send any data to Google, sending way more data to Avast instead."

So he reported his findings to Mozilla, who agreed, and immediately yanked all of Avast and AVG extensions from the Firefox extension repository. Avast have stated that they are working to correct this and will have an acceptable extension, which collects far less information, available as soon as possible.

Leo: Can you still use a Chrome extension? Is there a Chrome extension?

Steve: Yes. Google has been notified; and, as of this reporting, has not taken the same action. Google had not pulled the extensions from the Chrome Web Store. And my annoyance with this behavior is that companies are essentially doing everything they can get away with. This is not the end of the world for any of Avast's 400 million users. As we say, you know, they're sort of like, well, you know, you're getting what you paid for, which is you didn't pay anything.

But they appear to have been collecting all of this data just because they could. No one said no. No one had looked. And it's just not possible for every piece of communicating software to be examined and vetted ahead of time. I mean, Mozilla, they wouldn't have allowed this. But they just said, oh, you know, it looks like, you know, like click the button if you abide by our guidelines. And so Avast said, oh, yeah, we do. Well, when someone said, uh, take a look at this, Mozilla said, holy crap, and immediately yanked them all.

Unfortunately, it needs to be preemptive, rather than after the fact. And it'll be interesting to see how, if the clickstream is really curtailed, like they're clearly going to work to maintain as much of this as they can while still working to please Mozilla because they're clearly monetizing all of their customers' browsing habits. I mean, again. And if it were right upfront, if they were being really clear to their users that we're giving you free antiviral, and in return we're using anonymous collection of your browsing history and selling that somewhere else, it's like, then if the user says yeah, okay, fine, then okay. And I think most users probably would. Not our listeners.

Leo: Yeah.

Steve: Something that I ran across last summer and was very interested in and have on my list to get to in early 2020, is a new VPN, and by that I mean protocol, not like another version of OpenVPN. This thing's called WireGuard. And it pinged onto my radar this week because of an interesting announcement regarding it. So I have not yet made time to discuss it in depth. I will be. And I am not alone in believing that this is the future of VPNs. As I said, I first became aware of it last summer. After reading into it a bit, I bookmarked the project for follow-up, to tell our listeners. And I will be doing so.

Leo: We should make it clear, this is not a client you're talking about. It's a server. This would be a replacement for OpenVPN or...

Steve: Correct. It's not a client or server. It is, like, both ends. It is a...

Leo: Oh, okay, it's both. It's a protocol, so you'd have to have a WireGuard compatible client to use it.

Steve: Correct. It was everything I loved, a complete reconceptualization and a rewrite from scratch of a Virtual Private Network, very much as I was planning to do with CryptoLink, as our longtime listeners will recall. It's a simple take on the VPN. In their own words, they say: "WireGuard is an extremely simple, yet fast and modern VPN that utilizes state-of-the-art cryptography. It aims to be faster, simpler, leaner, and more useful than IPSec, while avoiding the massive headache. It intends to be considerably more performant than OpenVPN. WireGuard is designed as a general purpose VPN for running on embedded interfaces and super computers alike, fit for many different circumstances. Initially released for the Linux kernel, it is now cross-platform - Windows,

macOS, BSD, iOS, and Android - and widely deployable. It is currently under heavy development, but already it might be regarded as the most secure, easiest to use, and simplest VPN solution in the industry."

So sometime after that, while doing research for the podcast, I encountered an independent set of benchmarks which compared WireGuard with OpenVPN, and it wasn't even close. WireGuard blew the packets off OpenVPN. So I have not yet switched over to it myself, only because I've got a bit too much on my plate as it is. But it goes a long way toward solving one of the longstanding annoyances with a VPN, which is, as we know, it tends to be a little slower than a non-VPN connection.

So the next time I encountered WireGuard was a total coincidence. The sponsor of my OWASP SQRL presentation in Sweden was a company named Mullvad. I think their whole name is Mullvad VPN, which I'd never heard of at the time. And anyone who's seen the SQRL Sweden presentation video may have noticed some of the signage down there. Anyway, they took everybody to dinner the night before; and I got to meet the president and founder, and a bunch of the techies there. And I was really interested because they exclusively use WireGuard.

And they've got a really interesting model. When you sign up on their website, you get a serial number. That's it. No username, no email address, just a serial number. Then you somehow arrange to send them money. And they told some rather funny stories about receiving boxes of coins from anonymous people saying here's my money, and then here's the serial number. And so they fund that serial number, and you get to use WireGuard with multiple endpoints and so forth. Anyway, I ended up being very impressed with the company, met the owner, and they seemed like neat people.

So what happened this week is that Steven Vaughan-Nichols, writing for ZDNet, asked rhetorically: "How much are people looking forward to WireGuard, the new in-kernel Linux virtual private network?" Linus Torvalds recently said: "Can I just once again state my love for it and hope it gets merged soon?" He said: "Maybe the code isn't perfect. But I've skimmed it; and, compared to the horrors that are OpenVPN and IPSec, it's a work of art." And then Steven Vaughan-Nichols wrote: "That's not really damning with faint praise because, for Linus, this is high praise."

He said: "WireGuard has now been committed to the mainline Linux kernel. While there are still tests to be made and hoops to be jumped through, it should be released in the next major Linux kernel release, 5.6, in the first or second quarter of 2020." He said: "It's been in development for some time. It's a layer 3 secure VPN. Unlike its older rivals, which it's meant to replace, its code is much cleaner and simple. The result is a fast, easy-to-deploy VPN. While it started as a Linux project, WireGuard code is now cross-platform and is now available on Windows, macOS, BSD, iOS, and Android." And then I have some more details in the show notes for anybody who's interested.

He says - I skipped a bunch. "WireGuard works by securely encapsulating IP packets over UDP. Its authentication and interface design has more to do with Secure Shell than with other VPNs. You simply configure the WireGuard interface with your private key and your peers' public keys, and you're ready to securely talk." So that's all there is to it. He says: "When it arrives, I expect WireGuard to quickly become the new standard for Linux VPNs. With its tiny code size, high-speed cryptographic primitives, and in-kernel design, it should be faster than all other existing VPN technologies." WireGuard's not just fast, it's secure, as well.

With its support of state-of-the-art cryptography technologies such as the Noise protocol framework; Curve25519, which of course I'm a fan of, that's what SQRL uses; ChaCha20, Poly1305, BLAKE2, those are all Bernstein's works; and other stuff. Oh, and then I was surprised he mentioned, he said: "And this is why some companies - like

Mullvad VPN - adopted WireGuard long before it was incorporated into Linux. As Mullvad co-founder Fredrik Strmberg wrote two-years ago: 'We find WireGuard beneficial for a number of reasons. Its simplistic design in few lines of code makes it easier for sysadmins and developers to integrate it correctly, and harder for them to get it wrong.'"

He says: "Thus, WireGuard will move the world one step closer to our own vision of making mass surveillance ineffective." So anyway, I think we've got a new VPN here on the horizon, and I will be taking our listeners into a deep dive of it early next year.

Leo: That's cool.

Steve: Which brings us to VPN-geddon. Once again, we have breathless headlines. I have five of them.

Leo: It's nonstop.

Steve: "New Linux Vulnerability Lets Attackers Hijack VPN Connections." "Linux Bug Opens Most VPNs to Hijacking." "New Vulnerability Lets Attackers Sniff or Hijack VPN Connections." "New Linux Bug Lets Attackers Hijack Encrypted VPN Connections." And, finally, "Networking Attack Gives Hijackers VPN Access." None of which is true.

Leo: Hmm.

Steve: They all sound really bad. And again, I guess partly the problem is the guys who posted the advisory did make this sound like the end of the world. They kind of overhyped what it was they had done. But for our more technically oriented listeners, I think everybody will find this sort of interesting. But first, not to pick on any one of the publications, but every one of the publications that I routinely monitor for security news had this as one of their headlines and took the opportunity to say something about it.

So, for example, one of them in the press that people would be reading says: "A team of cybersecurity researchers has disclosed a new severe vulnerability affecting most Linux and Unix-like operating systems - FreeBSD, OpenBSD, macOS, iOS, and Android - that could allow remote network-adjacent attackers to spy on and tamper with encrypted VPN connections. The vulnerability, tracked as" - and we have a CVE number - "resides in the networking stack of various operating systems that can be exploited against both IPv4 and IPv6 TCP streams. Since the vulnerability does not rely on the VPN technology, the attack works against widely implemented Virtual Private Network protocols like OpenVPN, WireGuard, IKEv2/IPSec and more, the researchers confirmed.

"This vulnerability can be exploited by a network attacker controlling an access point or connected to the victim's network just by sending unsolicited network packets to a targeted device and observing replies, even if they are encrypted. As explained by the researchers, though there are variations for each of the impacted operating systems, the vulnerability allows attackers to" - and we have four bullet points - "determine the virtual IP address of a victim assigned by the VPN server; determine if there is an active connection to a given website; determine the exact sequence and acknowledge numbers by counting encrypted packets and/or examining their size; and inject data into the TCP stream and hijack connections."

Okay, now, nobody would be blamed, if you were to read that, for thinking, holy crap, this is really bad. Except it's not true. If it were, it would sound like VPN-geddon. But it's not. So that was mostly a repeat of the announcement of these so-called "security guys." I mean, they did some stuff. But they didn't really do what they claimed to have done. I've skipped the beginning of their opening disclosure because it was pretty much what I just read.

They said: "There are three steps to this attack: Determining the VPN client's virtual IP address." So oftentimes you connect up to a remote OpenVPN server, and you'll get a 10.8.0.something virtual IP because the VPN server is giving you a different IP that routes through your network stack to get to the virtual interface that the VPN client has established on your local machine. You talk to that, using that IP, and it goes to the virtual VPN interface, which then encrypts it and sends it to the physical interface, which then transits it out over the public Internet, like to the VPN, which then decrypts it and lets it loose on the Internet.

So step one, determining the VPN client's virtual IP address, not their physical IP address. Two, using the virtual IP address to make inferences about active connections. And then, three, using the encrypted replies to unsolicited packets to determine the sequence and acknowledgment numbers of the active connection to hijack the TCP session.

Okay. So they said then in their disclosure there are four components to the reproduction: the victim device connected to an access point; the access point, controlled by the attacker, meaning that all of the victim's traffic is transiting through the access point. So it's a man in the middle, basically. The VPN server, that is remote, probably, not controlled by an attacker. And then, although they do have the VPN server at 10.8.0.1, so they're apparently looking at the traffic going to the VPN server. And then a web server not controlled by the attacker, with a public IP out in the real world.

So they say: "The victim device connects to the access point, which for most of our testing was a laptop running create_ap. The victim device then establishes a connection with their VPN provider." And, okay, so this is sort of a special case because they apparently have access to the VPN server at 10.8.0.1, so they're also seeing the virtual traffic transiting the access point. So that's sort of a bit of a special case, but okay.

Then they said the access point, meaning the man in the middle, can then determine the virtual IP of the victim by sending SYN-ACK packets to the victim device across the entire virtual IP space. And they say the default for OpenVPN is 10.8.0.0/24. So it's 10.8.0.x, right, where "x" is between 1 and 254. So they said: "When a SYN-ACK is sent to the correct virtual IP on the victim device, the device responds with a reset packet. When the SYN-ACK is sent to the incorrect virtual IP, nothing is received by the attacker."

So, okay. So first of all, this is the definition of a kludge. So they're determining the virtual IP of the interface that the victim is using by essentially port scanning it by sending SYN-ACKs to some random port, probably doesn't matter, but at all of the different IPs available because they saw that, when they hit the right one, it will object with a reset, as it should, over TCP protocol. And if a SYN-ACK comes to just a random IP that nobody's listening on, it goes nowhere, so nothing happens.

Okay. So they're saying from their position they can get the virtual IP that the VPN server assigned to the client, who is the victim. It's like, okay. So then they say, and here it really gets a bit of a stretch: "Similarly, to test if there is an active connection for any given website, such as" - and they list as an example 64.106.46.56. Don't know what that is. "For example, we send SYN or SYN-ACKs from 64.106.46.56 on port 80 or 443 to the virtual IP of the victim across the entire ephemeral port space of the victim."

Okay. Now, the ephemeral port space starts at 1024. Remember that the service port space is defined as 1 to 1023. Those are the ones you need to be privileged. You need to be root in order to open a listening socket on those low port numbers. Apps like OpenVPN are able to do it in the high port space. And outgoing connections from like a web browser automatically walk up through that ephemeral port space. So they're sending SYNs or SYN-ACKs from a public IP to port 80 or 443 that they assume the victim has traffic with that public server on, again noticing a slight difference in behavior.

So they used the term "four-tuple," which we remember from when we used to talk about the TCP protocol. The four-tuple is the source IP, the source port, the destination IP, and the destination port. You have to have all four of those correct, and that defines the endpoints of a TCP connection. So they said: "The correct four-tuple will elicit no more than two challenge ACKs per second from the victim, whereas the victim will respond to an incorrect four-tuple with a reset for each packet sent to it."

So this doesn't let you determine the site that somebody is talking to. And remember there are four billion public IPs, give or take. But if you have to use the first scan to figure out the virtual IP of the person, and you wanted to see whether they were talking to a given public server, you would send - basically you would DoS this victim with TCP SYNs or SYN-ACKs on the port that you need to know also, that you assume they're connecting to this remote server over. So once upon a time 80, more likely 443, or who knows what if it's a different type of server.

So you send all these SYNs and SYN-ACKs as if they were from that public IP and port 80 or 443, to all of the ephemeral ports because you don't know what the outbound port number on the virtual IP is that the victim is currently using. So you spray them all. And you get a reset if you guess wrong, or no more than two challenge ACKs per second if you got the four-tuple right in order to confirm that this person in fact has an open TCP connection to the remote server. Notice that we're not able to see into the traffic; right? It's encrypted. The VPN is intact. It's still there. It's working. So what we're doing is, from this privileged man-in-the-middle position, we're sort of spraying the victim with stuff, TCP garbage, to learn what we can of their connection.

And then they say here: "Finally, once the attacker determined that the user has an active TCP connection to an external server, we will attempt to infer the exact next sequence number and in-window acknowledgment number needed to inject forged packets into the connection." Now, I'll just note that they can't get the private key which is being used to encrypt these packets. So you can't forge a packet. I mean, you can send nonsense to the victim. Okay. But so what?

Anyway, so then they say: "To find the appropriate sequence and ACK numbers, we will trigger responses from the client" - we're going to send more crap to it - "responses from the client in the encrypted connection found in part two." So they're acknowledging that it's an encrypted connection. So we're going to spray the encrypted connection with some more crap and see what bounces off of it. They said: "The attacker will continually spoof reset packets into the inferred connection until it sniffs challenge ACKs. The attacker can reliably determine if the packets flowing from the client to the VPN server are challenge ACKs by looking at the size and the timing of the encrypted responses." Again, they can't read them, but they're going, ooh, look, it's tiny, so it must be an ACK.

"In relation," they say, "to the attacker's spoofed packets. The victim's device will trigger a TCP challenge ACK on each reset it receives that has an in-window sequence number for an existing connection. For example, if the client is using OpenVPN to exchange encrypted packets with a VPN server, then the client will always respond with an SSL packet of length 79" - that is, 79 bytes - "when a challenge ACK is triggered."

They say: "The attacker must spoof resets to different blocks across the entire sequence number space until one triggers an encrypted challenge ACK." Now, as we recall from TCP, the sequence number and ACKs are 32-bit numbers. And so, and we don't know how large the window is, so we're having to spray the possible sequence number space, the 32-bit space, with this junk, looking for one that triggers an encrypted challenge ACK. And again, it's encrypted. They said: "The size of the spoof block plays a significant role in how long the sequence inference takes" - yeah, no kidding, because it's 34 billion - "but should be conservative" - uh-huh. You need to do a lot of them. You can't assume the blocks are too big - "but should be conservative as to not skip over the receive window of the client." You've got to hit the window in order to get an ACK back.

"In practice, when the attacker thinks it sniffs an encrypted challenge ACK, it can verify this is true by spoofing X packets with the same sequence number. If there are X encrypted responses with size 79 triggered, then the attacker knows for certain it is triggering challenge ACKs," and he says, "at most two packets of size 79 per second." Anyway, I'm not going to continue with this nonsense. Everyone gets the idea. This is just such a crock.

The first parts of this so-called "attack" attempt to simply probe the state of the TCP stack by flooding its IP and port space with packets designed to elicit a reply. Then they spoof packets from an assumed public IP space designed to, again, elicit a reply. And all they're doing is they're seeing that basically they're annoying the virtual interface which is encrypted by whatever VPN is in use. They're annoying it, and the TCP stack is responding as TCP stacks do to their poking at it with encrypted results that they can't read. And the real downfall is that then they say that they are able to - what was the term? I can't even - my brain won't remember it, it's such nonsense. Oh, that they're able to hijack the connection. Except it's encrypted.

And so, what? I mean, it's true that a TCP connection can be hijacked if it's not encrypted because the attacker can see the sequence numbers going back and forth. And we talked a long time ago about how BGP routers, routers that were using Border Gateway Protocol in order to keep their routing tables synchronized, there were attacks on their long-term persistent TCP connections by people who from outside were blasting them with nonsense and guessing what the sequence numbers were, and in fact being able to hijack the connection. Which works if it's not encrypted. But if it's encrypted, you can't hijack the connection. I mean, again, all you can do is piss off the TCP/IP stack on the virtual interface and cause it to emit a bunch of encrypted annoyed ACK packets. But that's it.

So anyway, I just thought this was interesting because it was an example of a complex attack which sounds really bad, but it's probably like the people who know better are like, yeah, okay. So? And admittedly, it does breach some of the privacy guarantees that you would like your VPN to be permitting, but only if you establish a man-in-the-middle position, and you have the ability to also see the virtual IP traffic going to the VPN server. So that's a bunch of necessary preconditions. And even so, you are never able to see into the traffic. And you can't even see the remote server. It's talking to you. You have to guess from the four billion public IP space and send probes at the poor virtual interface and see if it annoys it or not, and in which way it annoys it. Anyway, I just thought, wow, okay. Game over.

Leo: It feels like the authors of this article were a little disingenuous, like they didn't have anything here.

Steve: Yeah. They really didn't have anything.

Leo: So what was the point?

Steve: They really overplayed this. I just - they were amateurs who thought, ooh, wow.

Leo: Look what I can do, yeah.

Steve: Look what I can do, yeah. And it's like, yeah, okay.

Leo: So what?

Steve: Unfortunately, yeah, the idea of hijacking a VPN connection, that has, I mean, those words have meaning. And they've abused the meaning.

Leo: Can they break the connection?

Steve: Yeah, they could probably bring it down, yeah. Maybe.

Leo: Okay, yeah.

Steve: Yeah, I mean, if they got lucky, well, see, you'd have to send a reset. And you can't send a reset because you don't have any idea what the encryption is because you'd have to send an encrypted reset, and they can't generate any valid encrypted packets. So I don't really - I'm not even sure, I mean, maybe if you flooded it with enough nonsense you could just annoy the...

Leo: That's just a DDoS. Yeah, that's not - yeah.

Steve: Yeah, exactly, just a DDoS. And so that's not special. Yeah, I mean, they said - that fourth point, I'm looking at it. Inject data into the TCP stream and hijack connections.

Leo: But you can't.

Steve: Okay, well, right. Not meaningful data. You could just inject nonsense.

Leo: And I wouldn't see it anyway because it doesn't fit the schema; right?

Steve: Right, right. It bounces off the leading edge of the TCP/IP stack. So it's just like, nah, just going to be ignored.

Leo: Oh, yeah. Maybe they just - maybe they weren't very sophisticated themselves, and they thought that [crosstalk].

Steve: I think they probably weren't very sophisticated.

Leo: Didn't have any impact at all.

Steve: Got a lot of press. Got a lot of press.

Leo: Yeah, well, and you know, to be fair, most of the press, even the tech press isn't technically that technical. And so they're just going, oh, oh, oh. So it's good we've got you. That's all I can say.

Steve: Well, and that's why I didn't read the beginning of their disclosure, because it was echoed by the tech press.

Leo: Right.

Steve: Who just copied this alarming notice and said, "Oh, no."

Leo: Right.

Steve: Anyway, "Oh, no" is right.

Leo: Oh, no. No. No geddon of any kind. Steve Gibson is our man. He is at GRC.com. That's the Gibson Research Corporation. SpinRite lives there, the world's best hard drive recovery and maintenance utility. He's working on the new one. And given a little time, I think he will come up with something pretty special. But you can get the existing one right now and get a free upgrade when it's out, 6.1 is out.

You can also go there to get the show. He's got 64Kb audio, just like we do. But he also has 16Kb audio for the bandwidth-impaired and a full human-written transcription, so you can read along as you listen, all at GRC.com. So check that out. He's on the Twitter at @SGgrc. And you can direct message him there if you've got comments, questions, suggestions: @SGgrc.

We have audio and video of the show. Holy cow. Why would you want video? I don't know. Because there's pretty pictures. All you've got to do is go to TWiT.tv/sn. You can also watch on YouTube. And do, if you will, subscribe. That way you'll get a copy the minute it's available of a Tuesday afternoon. We do the show usually 1:30 Pacific, 4:30 Eastern, 21:30 UTC, if you want to watch live, TWiT.tv/live. You can chat live, too, at irc.twit.tv.

Steverino, thank you so much. I'll see you next week on Security Now!

Steve: Right-o.