Rowhammer's Nine Lives

Description: This week we're going to note an urgent vulnerability created by an add-on to Atlassian's Confluence corporate workgroup server. Next week's USENIX Security Conference will be presenting TLS-Anvil for testing TLS libraries. Google has decided to again delay their removal of third-party cookies from Chrome, and attackers were already switching away from using Office Macros before Microsoft actually did it. We have a bunch of listener feedback, some thoughts about computer science theory and bit lengths, and some interesting miscellany. Then we're going to look at the return of Rowhammer thanks to some new brilliant and clever research.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-882.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-882-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We will talk about TLS-Anvil, a really interesting testing framework for TLS libraries, works really well. Google is delaying - who could have predicted it? - once again the removal of third-party cookies from Chrome. And proof positive that turning off macros in Microsoft Office really works. Plus a look at Rowhammer. It's back again, the attack that just never quits. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 882, recorded Tuesday, August 2nd, 2022: Rowhammer's Nine Lives.

It's time for Security Now!, the show where we cover your security, privacy online with this guy right here.

Steve Gibson: All that.

Leo: And all that jazz. Mr. Steve Gibson. Hello, Steve.

Steve: Yo, Leo. Great to be with you. We're at Episode 882. The next one of note will be in six weeks. It'll be 888, so just because that's a fun number. But that'll, you know, here we're in August already.

Leo: Already in August. Man.

Steve: Okay. So this is not the actual name of the exploit. The name of the exploit didn't really move me much. It was called "Half-Double." But what it is...

Leo: Half double is a single.

Steve: Yeah, exactly. But it is brilliant. And so I just thought it would be too fun not to share it with our listeners. So today's title is "Rowhammer's Nine Lives."

Leo: Oh, man.

Steve: Because, yes, it isn't gone yet. And these guys have figured out a way to still flip bits in DRAM and exploit it. In fact, I don't mention this until the very end. But on a brand new, completely up-to-date Chromebook, an unprivileged user can obtain root, kernel access in 45 minutes.

Leo: Oh. On a Chromebook.

Steve: On a Chromebook. So of course this has all of the cloud people terrified because they're all sharing hardware assuming that you've got protection against a random virtual server in the cloud obtaining root on its host. And this makes it possible. Anyway, we're going to have fun talking about that. But we're going to first note an urgent vulnerability created by an add-on to Atlassian's Confluence corporate workgroup server. Next week's USENIX security conference will be presenting not only the subject of the show, this new Rowhammer attack, but also something very cool called TLS-Anvil, which is used for testing TLS libraries. And 13 of them were tested. We'll talk about what it found. And it's like, it's not nothing.

Google has decided again, although I'm on their side on this, well, at least I'm on their side on the initiative, they've decided to again delay their removal of third-party cookies from Chrome, and for presumably good reasons. We'll talk about that. And it turns out that attackers were already switching away from using Office Macros before Microsoft actually did it. We'll talk about the back story there. We also have a bunch of listener feedback, some thoughts about computer science theory and bit links that I thought our listeners would also find interesting. Some interesting miscellany, and then we're going to look at the return of Rowhammer which we never really seem to get rid of, thanks to some really just awesome brilliant and clever research. And of course I would argue that this is one of our better Pictures of the Week. We've had...

Leo: People are howling over this thing.

Steve: ...some that are really fun. I gave it the title. It didn't come with a title, but I like the title. And anyway, it's a great one. So I think another overall great podcast for our listeners.

Leo: Going to be a lot of fun. We'll get to the Picture of the Week. All right. Picture of the Week time, Steve.

Steve: Okay. So this is two captures from Star Trek, the original series. And I titled this "Keptin, we cannot go to warp." So the first one is from Kirk's vantage point looking out across Zulu on the left and Chekov on the right and to the view screen at the front of the bridge, where we see the little what I call the rollercoaster dots, and a blue screen, the little spinner saying "Working on updates: 30%." And so we see Chekov sort of like turning his head back to look at Kirk, presumably saying, "Keptin, we cannot go to warp."

And the second frame is a very unhappy-looking James Tiberius Kirk, kind of like growling because he wants to go to warp, and no. His ship will not let him. And I will note, we'll be talking about this later in Miscellany, but Uhura back there, ever present on the bridge through all of the original series and all six movies, as well. So anyway, yes, James is not happy.

Leo: [Growling]

Steve: That he's having to wait for his starship to update. And lord help us if Windows is embedded into future starships. Please don't let that happen. Everybody has always wanted me to write a Windows operating system, Leo.

Leo: Oh, no.

Steve: I would.

Leo: Come on.

Steve: Well, I mean, I'm not going to. I've said no many times. But just like imagine if Steve wrote Windows. Well, I'm not going to do that. But okay. If anyone tried to stick it into a starship, okay, I would write that operating system. I'm just saying, if that's what it takes not to have updates ever, then fine, I'll step up.

Okay. So the Australian software company Atlassian offers something called Confluence, which is a web-based corporate Wiki, written quite a while ago, 18 years ago, back in 2004, in Java.

Leo: We use it, actually. And have for years, yeah, yeah.

Steve: Oh, cool. It's one of Atlassian's more popular products. It's packaged as a standalone turnkey that includes a built-in Tomcat web server. And the database is HSQLDB, H-S-Q-L-D-B, which is also a Java-based database, so it makes sense that that's what it would choose, although other databases can be used with Confluence. It's modern collaborative software. It's used to help teams collaborate and share knowledge. In a corporate setting it's used, for example, to capture project requirements, assign tasks to specific users, and manage one or more calendars. It's typically used on-premise all over the place, both in corporate enterprise and government environments.

Leo: And now I'm waiting for the other shoe to drop. Is there a...

Steve: That was the good news.

Leo: Yeah.

Steve: We're talking about it today because it appears to have recently come to the attention of ever-prolific attackers who, based on the troubles it's been having recently, it's come to their attention because it's been, you know, vulnerabilities have been surfacing. And unfortunately, as with any product that was first written 18 years ago, before today's hyper-attack-aware climate existed, some of its coding practices from back then appear to be being turned into vulnerabilities.

Now, although Confluence has recently suffered, as I said, from a series of troubles, it popped up again just last week due to a very serious problem with the installation of, believe it or not, default hard-coded credentials. As we know, any time you hear that a problem has been created by the discovery of default hard-coded credentials, meaning a username and password with privileges no one should have, by default being part of an installation, it's difficult to chalk that up to anything other than some previous very bad decision.

In this instance, the trouble arises from the installation of an app called Questions for Confluence. So it's not the core Confluence server itself, it's an add-on app. It's often installed on Confluence Server and Data Center on-premise instances because it allows employees to ask questions and receive answers from a company's various internal departments. The installation of this Questions for Confluence app creates a user named - the user's name is "disabled system user" with the username "disabled system user" and the email of "dontdeletethisuser@email.com." So, you know, a fake email placeholder, but "disabled system user" as the name of the account and the name of the user. And unfortunately, contrary to the name given to the account, it's not at all disabled, although it is indeed a system user.

Atlassian released a patch that disables this built-in hardcoded account 13 days ago, on July 20. However, Confluence server admins did not have much time to install Atlassian's fix because, through no fault of Atlassian's, the system account username and credentials for this account were posted the next day to Twitter.

Leo: Of course.

Steve: Uh-huh, by someone using the handle "annoyed researcher."

Leo: Oh.

Steve: Okay. I don't know the back story for that, but it's uncool, right, to do that under any circumstance. I don't care how annoyed somebody is, that's just not the way it's done. In fact, I would argue you cannot call yourself a researcher, no matter how annoyed you may be, if you're posting credentials for a just-released flaw that no one will have had time to fix.

Leo: Yeah. You have to do responsible disclosure. That's just - everybody has to do that, yeah.

Steve: So as things go these days, it didn't take the bad guys long to put "annoyed researcher's" information to nefarious use. The cybersecurity firms, both of them, GreyNoise and Rapid7, both reported seeing immediate and continuing exploitation of this, I guess we'll call it a vulnerability, although it's hard to chalk up hard-coded credentials. I guess it is a vulnerability. It's not like they broadcasted it. But still, shouldn't be done. It's the kind of thing you do by policy rather than by mistake. I guess that's the problem I'm having. Anyway, within days of the release of the patch it was being used in attacks. CISA has stepped up and urged Confluence server owners to check to see if the vulnerable app has been installed on their servers; and, if so, to install Atlassian's patches.

And Atlassian noted that disabling the app, due to the nature of the problem, doesn't fix the issue because it's the app's installation that created the problem by creating this instantiation of this non-disabled disabled user. So Confluence server owners must either install the security fixes or manually disable that hard-coded "disabled system user" account which was created by this Questions for Confluence app.

So anyway, because this thing, I mean, I've been seeing Confluence problems pass by, and they just haven't risen to the level of bringing them to our listeners' attention. But this one really seemed like it was important enough that you just want to make sure that, if you're using this thing, that you've responded to Atlassian's security fix and/or killed or disabled that user.

And in fact Kevin Beaumont, the true security researcher, who tweets as @GossiTheDog, he said: "I would recommend Confluence be placed behind a VPN or reverse proxy," and he says, "for example, Azure App Proxy," he says, "requiring authentication." He said: "It's simply too historically vulnerable to leave online. You're a sitting duck," was his tweet.

And, you know, it's difficult to argue with Kevin's logic. Everyone knows that I think this should be standard practice now for all publicly exposed services. The rule would be don't publicly expose anything that doesn't utterly and absolutely, by definition, need to be publicly exposed. So what does? Web servers, obviously, need to be publicly exposed by definition. So does email, and probably DNS, although I even managed to not have my DNS publicly exposed. I use Level 3's big iron DNS servers as slaves to my master. So the DNS, the only DNS that my system allows is zone transfers from my two designated name servers for GRC.com, which are Level 3 monster DNS servers. And who knows? They're probably virtualized and distributed and not even actual single machines anymore.

But anyway, so I guess the point is it is possible to absolutely protect all of an enterprise's stuff by putting them behind, and this is what Kevin is talking about, an additional layer of authentication. Put them behind a VPN. Put them behind some additional boundary that requires authentication, and only then are you able to get to it. Or use an overlay network, and stick them on an overlay network so that the people have to authenticate to that in order to get there. But just don't have them as a public IP and port that Shodan is going to index so that the bad guys can jump on whatever it is the instant a problem is found. I mean, that's the model today.

So the only sane response to today's reality is, no matter what it is, no matter how secure you think it is, give it additional protection. It has to be the way we roll. And I've been following my own advice from like before we started talking about this.

During next week's 31st USENIX Security Conference being held Wednesday through Friday the 10th through the 12th in Boston, Massachusetts, a team of security researchers will be unveiling what they call "TLS-Anvil." And I guess the point of the

name is to use this TLS Anvil to forge fully secure and compliant TLS connection libraries because that's what it's designed to test.

In the description of their forthcoming talk, which as I said they'll be giving at the end of next week, they explain. They said: "Although the newest versions of TLS are considered secure, flawed implementations may undermine the promised security properties. Such implementation flaws result from the TLS specifications' complexity, with exponentially many possible parameter combinations. Combinatorial Testing," which they abbreviate CT through their work - "is a technique to tame this complexity, but it is hard to apply to TLS due to semantic dependencies between the parameters, and thus leaves the developers with a major challenge referred to as the 'test oracle problem,' which is determining if the observed behavior of software is correct for a given test input.

"In this work," they write, "we present TLS-Anvil, a test suite based on Combinatorial Testing that can efficiently and systematically test parameter value combinations and overcome the oracle problem by dynamically extracting an implementation-specific input parameter model that we constrained based on TLS-specific parameter value interactions. Our approach thus carefully restricts the available input space, which in turn allows us to reliably solve the oracle problem for any combination of values generated by the Combinatorial Testing algorithm."

And finally they say: "We evaluated TLS-Anvil against 13 well-known TLS implementations, including OpenSSL, BoringSSL, and NSS," Netscape's Security Suite, the original one. "Our evaluation revealed two new exploits in MatrixSSL, five issues directly influencing the cryptographic operations of a session" - which they refer to as handshakes, at the handshake level - "as well as 15 interoperability issues, 116 problems related to incorrect alert handling, and 100 other issues across all tested libraries." So needless to say, this was worth doing. And it is just so cool that we have a world where academics are able to tackle something this complex, all of this mumbo jumbo about combinatorial testing and input parameter modeling and/or testing oracles. The point is, we could reduce all that to "Testing TLS is hard. We did it anyway."

Their paper is really interesting. I've got the link to it in the show notes for anybody who is interested. The 13 libraries that they tested were: BearSSL, BoringSSL, Botan, GnuTLS, LibreSSL, MatrixSSL, Mbed TLS, NSS, OpenSSL, Rustls, s2n, tlslite-ng, and wolfSSL. As it happens, through the years we've talked about most if not all of these. They're all familiar to me. I'm sure they are to our listeners, those who have been with us the longest, at least, and recall, for example, that "s2n" is Amazon's from-scratch open source implementation of TLS. They wrote it because of AWS. They needed a TLS for their own cloud services.

But they observed that OpenSSL, as we've often observed, had grown in size to half a million lines of code, 70,000 of which were tied to TLS processing. But they could be replaced, and they did replace them with about 6,000 lines of code, which Amazon observed is far easier to audit and to actually have some sense of knowing what the code is actually doing. How can you do that with half a million when they're all tangled up with each other?

So anyway, this chart which you had on the screen earlier, Leo, thank you, in their paper gives us a cross reference to those 13 TLS proposed libraries with the number of different type of exploits found, problems with the handshake crypto, problems affecting client and server interoperability, the generation of alerts, which is to say in the protocol when either end does something illegal, the other end is supposed to send an alert and disconnect. It's like, you're bad, and I'm hanging up on you. And then other problems.

Their paper was 18 pages. I'm not going to go into it in great detail. But they definitely found problems of varying degrees in every one of the 13 TLS libraries. And looking over

the chart, it appears that the best libraries were BoringSSL, Botan, NSS, OpenSSL, and Amazon's s2n. Oh, as well as tlslite. And these guys now really understand the state of play with the industry's TLS support. They wound up having, like, creating the tool, running the libraries against it, and then having to sit back and ask themselves, okay. It's not exactly what we expected. We didn't know what we were going to get. How do we decide to think about this?

And they ended up defining two degrees of success for this test. They wrote: "A strictly succeeded test means that a library behaved exactly as expected. If multiple test cases are performed during the execution of a test template, the system under test must have behaved correctly across every one of them." But then they also created a softer, gentler definition. They said: "A conceptually succeeded test means that an implementation did not precisely fulfill the RFC requirements, or did not do so in all test cases, but effectively behaved correctly." And I'll explain what happened in a minute because when they had this thing, they then, to be responsible, they needed to talk to the implementers. And it's actually the case that the implementers convinced them that it was necessary not to be strict. So they said, okay, let's just be conceptual, then. That's good; right?

So they said, for example: "This usually applies to tests where a fatal alert was expected, but the library either only closed the connection, but did not send an alert first; or the alert description did not match the RFC's specification." So we should understand that these guys implemented TLS-Anvil strictly from the RFC's. That was their starting point was what does the spec say the library must do in each and every case? And the act of testing that, I mean, we don't know what internal tests TLS developers may have. But if we presume that they've been using their internal tests, and how could you not, then this TLS-Anvil's results suggest that in at least some cases those internal tests were wanting.

So they then give some examples of what they found. They said: "We count all tests as 'passed' that either succeeded strictly or conceptually" - and now we know what that means - "and include the percentage of passed tests. We further list the ratio of conceptually to strictly succeeded tests as an additional metric to compare how close an implementation is to the RFC for our tests. Rustls, for example, passed many tests, but most of them only succeeded conceptually, as Rustls often did not send any alerts." And again, it's like, you're supposed to send an alert to tell the other end why it just misbehaved. But Rustls just said, you know, eff off, and disconnected. Which it's been argued...

Leo: It's better than nothing.

Steve: It gets the message across. And "Botan, in contrast, often fulfilled the expectations," they wrote, "of our tests, and at the same time was very accurate with alert descriptions."

Leo: Good, yeah.

Steve: Yeah. They said: "We generally found that most libraries pass a high ratio of the test templates, with NSS, BoringSSL, tlslite-ng, and OpenSSL passing around 97%" - which I think is really good - "of their applied server tests. Among the client tests" - okay, so of course "server tests" meaning that you're using the library to accept connections, "client tests" meaning you're using the library to generate new TLS connections to a remote server. So they said: "Among the client tests, BearSSL, BoringSSL, and Botan have the highest ratio of passed tests with 97.3%, 96.8%, and 96.2%, respectively."

They said: "We expand upon the results of libraries with significantly worse ratios, listing how many test templates of an RFC passed and how many were executed for each library. We grouped the results of test templates based on similar error cases and identified a total of 239 issues. We further categorized these findings based on their impact, and determined that three immediately led to exploits in wolfSSL and MatrixSSL." In other words, exploitable problems in those libraries. They're gone now, so long as you've updated recently, because these guys were doing all of this work before they went public and gave everybody plenty of time to fix.

They said: "Additionally, we found five issues affecting the cryptography of a handshake. As an example, the clients of MatrixSSL, Amazon's s2n, and wolfSSL are willing to negotiate parameters they did not offer." Which is interesting and not good. And they said: "While none of the parameters negotiated are sufficiently weakening the security to pose an immediate threat now, parameter negotiation is a basic security property of every cryptographic protocol to prevent current and potential future attacks." So again, another nice thing that these guys checked which the authors of MatrixSSL, s2n, meaning Amazon, and wolfSSL, missed. So now they know, and presumably they fixed that.

They said: "We identified 15 issues affecting the interoperability to an extent where a peer that operates within the boundaries of the RFC may not be able to complete a handshake." They said: "Note that this may also include intentional deviations by the developers if they break the implementation's correctness in regards to the specification."

They said: "100 issues account for various likely uncritical cases where a library deviated from the RFC beyond alert codes and where interoperability should not be affected. Examples of these findings are a bug in OpenSSL, which allowed multiple TLS 1.3 HelloRetryRequest messages, which can keep the client in a handshake loop." Okay, again, like technically a violation of the RFC, but nothing to worry about. But still, what's interesting about these is, if that isn't what the developers intended, then getting them to look at why that happened could reveal some other side effect of this. So always good to at least be sure you know why you deviated from the intended behavior.

They said: "Finally, we grouped 116 cases where a library did not send an alert or sent a different alert than requested by the RFC. These are minor deviations from the standard. However, in the past, information gained from the type of alert sent by an implementation has been used to mount side-channel attacks." They say: "To avoid such deviations, great care must be taken when designing the alert handling of an implementation. We hence chose to include these findings in our report to the developers." Meaning that they then, after creating this, produced a comprehensive report to each of the 13 developers of these packages.

And what they had to say about responsible disclosure and feedback I thought was very interesting. They said: "We responsibly disclosed all of our findings to the respective developers. During the disclosure process, multiple developers stated that they intentionally violate RFC requirements in specific cases. As an example, in TLS 1.2, peers are not allowed to resume a session that has been terminated by a fatal alert. However, when multiple sessions take place in parallel, this requirement is difficult to implement." Okay. So what I loved about that was that the ivory tower developers of the RFC said, if a peer terminates with a fatal alert, then you're not allowed to resume a session. But the guys who implemented this, or attempted to, said, okay, what do we do about multiple parallel sessions? You didn't say. You didn't tell us what to do. So these guys went one way that the TLS-Anvil guys took issue with.

They said: "Multiple developers also stated that they intentionally sent different alerts or no alerts at all." Intentionally. "One reason was to minimize the risk of creating an alert-

oracle for attacks." And these guys themselves mentioned that in the past the alerts that were sent had been used, had been leveraged in side channels. However, now they say: "We do, however, stress that the specified alert handling of current TLS RFCs does not result in a known exploitable oracle, but is considered to be secure." So they're saying it's not a good excuse.

They said: "Our original test suite contained 18 additional test templates, which we removed from the test suite after discussions with different library developers. Their reasoning convinced us that in these cases our interpretation of the RFC was too strict and that their library behavior was indeed valid." Very cool.

So they said: "Our presented evaluation does not contain these additional test templates anymore. There were some cases where the developers argued that it is unreasonable to follow the specification. For example, in some tests, a server that supports both TLS 1.3 and TLS 1.2 would need different alert handling for the same effective test. The situation becomes even more tricky when the server has not decided which protocol version to speak yet." Right? So again, two different protocol versions require different alerts to respond differently with alerts, depending upon the problem. Yet at this stage in the handshake, the version that's being negotiated hasn't been determined. And it could be either of them. So again, I love the fact that these guys were so insanely rigorous that these, I mean, they're edge cases, clearly. But, I mean, they're like an edge case's edge case. Yet they came out as a consequence of this.

So, they said: "For example, a server that receives a malformed or illegal ClientHello message would first need to evaluate the supported version of the client to decide upon the correct alert handling rules. Correctly handling these nuances," they said, "can be complex, and it is arguable whether the strict RFC conformance across all supported versions is worth the added complexity."

And they finished: "The security bugs we reported and most of our other reports have been acknowledged by the developers and will be considered for future releases." None were run around screaming immediately patch or they would have been done already. "Since most failed test templates only failed for single or very few libraries, we conclude that the developers in general share our understanding of the RFCs." So, just so cool. And I guess another way to read this would be ivory tower RFCs meet real world implementations. There are some places where it just doesn't make sufficient sense to go to all the trouble that absolute slavish adherence to the RFCs would require.

Anyway, all this beautiful work is on GitHub. The site is tls-anvil.com, which offers Dockers to allow the use of these tests by anyone. And I'm just - I salute this. This is precisely the sort of work that we need, and as much of it as we can get. So bravo to these guys. Very, very cool work. And it now exists. So if there's a 1.4, then the test suites can be augmented to incorporate that. This ends up being a benchmark reference against which TLS suites can be tested. And I just - that's so cool.

Google, well, last Wednesday...

Leo: They keep putting this off, don't they. They just...

Steve: They do.

Leo: They keep putting it off.

Steve: And I guess I'm sympathetic because I recognize, we all should recognize, how difficult change is. If there's one of the overriding themes of the podcast's lessons, it would be "Change is hard, and nobody wants it." Just say IPv6, and you can rest your case.

Anyway, their VP of the Privacy Sandbox initiative, Anthony Chavez, said last Wednesday: "The most consistent feedback we've received is the need for more time." Right, it's not Google, no no no no, they're ready to pull the plug.

Leo: Oh, yeah.

Steve: That's right. "More time to evaluate and test the new Privacy Sandbox technologies before deprecating third-party cookies in Chrome. This feedback," he wrote, "aligns with our commitment to the CMA" - we'll get to that in a second - "to ensure that the Privacy Sandbox provides effective, privacy-preserving technologies, and the industry has sufficient time to adopt these new solutions."

Now, okay. I would argue that the industry will not adopt anything until you pull the plug on the old solutions, and you've said, we told you we're going to pull the plug. We gave you plenty of time. We're sorry that cookies are gone, and you can't track anybody anymore. Here's the API that we created to let you do that. So start doing it. Again, it's like, okay. The lesson we keep seeing is you have to make people change.

Okay. So anyway, let's back up a bit. Google once again delayed its plan to terminate Chrome support for third-party cookies is what this amounts to. They first unveiled this privacy sandbox initiative back in 2019, when my moustache was a little less gray, with the announcement that its implementation would begin this year in 2022. Meaning that this would be the only way to do, what do we call it, interest-based advertising or something. I'm sure there's a term. Anyway, last year, after being scrutinized by the U.K.'s Competition and Market Authority, that's that CMA that Anthony referred to, and the U.S. DOJ, our Department of Justice, Google announced their intention to delay their third-party cookie phase-out until the middle of next year, so 2023.

And now, last Wednesday, what Anthony is announcing is another delay saying that now it won't be - I saw August of 2024 somewhere. But, you know, basically they won't end, they're promising not to end support for third-party cookies before the second half of 2024. And again, my feeling is, just do it. Just, you know, have the system in place. Have it well tested. Have it working. And then end the way it used to be, and make people switch. And, you know, in two years - are we still going to be here in two years? I think we are, Leo, yeah. So we'll get to see what happens in the second half of 2024. We'll be right here.

Under its current timeline, Google will expand the availability of its Privacy Sandbox trial to what they said was "millions of users globally," meaning that Chrome will begin to get this technology. And they're saying by early next month, so early September. Millions of users using Chrome will have what we've talked about before. The Topics API will be live. Then they plan to gradually roll out the test to more and more users throughout the rest of 2022 and 2023. They hope to officially launch the Privacy Sandbox APIs by the third quarter of 2023. Meaning, okay, about a year from now. Anthony said: "This deliberate approach to transitioning from third-party cookies ensures that the world can continue to thrive" - thrive.

Leo: The world? You mean the advertisers.

Steve: Yeah, uh-huh, "without relying on cross-site tracking identifiers or covert techniques like fingerprinting." Okay. So on the podcast we've talked about and we've been, if you'll pardon the choice of words, we've been tracking Google's proposed tracking technologies closely. The first attempt at replacing third-party cookies was their awkwardly named Federated Learning of Cohorts, you know, FLoC, which, Leo, you grabbed the fact that that was about birds, which...

Leo: All of the codenames in that generation were bird-focused.

Steve: Right, right. Then when that failed to take off, largely because no one understood how it worked, and it was way too opaque, Google announced their new approach called Topics. Which is largely the same, but is far more understandable and inherently transparent in the way it works. And as the name suggests, and we did a podcast on this, so we've completely covered this, Topics more transparently, you know, it works more transparently and makes it obvious how and what interests it is sharing with websites that users visit. It's got about 300 different topics. And when a user visits a website that requests the information, Chrome will offer from a pool three different topics in a random sequence that it knows that the user is interested in. So it's sufficiently cloudy, fuzzy, that you can't track a user by seeing what topics their browser is offering from one site to the next.

The site's publisher, then, that is, the site's server that receives these three randomly selected from a larger pool that the user has shown their interest in by virtue of where they've gone, to the sites they've visited in the past, that server can then choose to share that information with their advertisers, which they certainly would.

Okay. And being a technology hound, I understand and love these initiatives from Google. They make sense to me. But the big question in my mind will be whether any of this can actually replace all other side-channel tracking. As we've observed, if you turn off cookies, bad guys will just - or gray guys will just use some other means of tracking, like fingerprints, all of the stuff that the Panopticlick site shows is leaking from our browsers. I think that the only way we're ever going to be free of tracking altogether is through legislation, which makes it flatly and permanently illegal. Of course we'll first need a privacy protecting replacement, which Google's Topics does offer. But once that's in place, we're going to need governmental regulation, probably led by the EU, since they seem to have a penchant for that, to completely outlaw any and all other forms of anonymous Internet user identification, aside from the use of the sanctioned API which will hopefully become an industry-wide standard, and I think has the chance to.

So again, it's going to - everybody, every step of the way, this is going to require people being forced to change. So far it's already been slow and painful. I don't see any reason for that to be any different. Google's got this thing in place. They've got to stop moving the goalposts, or no one's ever going to believe that they're going to be serious. So hopefully, you know, latter half of 2024 we'll finally see Chrome remove third-party cookies completely, altogether, from the Chromium core. And by that time topics will have been around long enough so that, if somebody says, wait a minute, how are we supposed to track people? It's like, well, you're supposed to use Topics, and it's been around for a couple years. So it's your fault if you're not.

Leo: It's going to be a race because that's about Episode 990, thereabouts. It'll be a race between whether...

Steve: Wait, we have our - oh, yeah, I was just going to say we have our birthday on August 19th, but that doesn't correspond with Episode 999 because...

Leo: Fortunately, it does not.

Steve: Right.

Leo: Thank goodness. But yeah, it's going to be a race between whether Google clobbers third-party cookies or you clobber Security Now!, which will happen first. No one knows.

Steve: Aww.

Leo: Yeah, you can back out, Steve. It's not a permanent, you know. I'm glad to hear you saying that. No one wants you to stop.

Steve: Okay. Well...

Leo: All right.

Steve: We'll see how it goes. May have to hang in, well, no, I'm not making any promises.

Leo: No, don't say anything, just we'll get there.

Steve: Don't say anything.

Leo: We'll see what happens when the time comes.

Steve: Okay. So Proofpoint released the results of their email-based malware attack analysis. In their report, they observed that their analysis of campaign threats, meaning malware campaigns, which include threats manually analyzed and contextualized by Proofpoint's threat researchers, they found that the use of macro-enabled attachments by threat actors had decreased approximately 66%, meaning it's dropped to one-third of what it was before, between October 2021 and last month, June 2022. And remember that Microsoft previously announced it would begin to block XL4 and VBA macros by default for Office users in October of 2021 for XL4, which coincides exactly with the beginning of the observed drops in their use.

Leo: It works. It works.

Steve: Yes, yes. And also in February of this year for VBA. And as we know, there was some confusion last month by Microsoft saying, oh, we're not sure. Okay, now we are

again. But now it's going to just be a documentation change. The disabling has ultimately remained in place, thank goodness. So not surprisingly, threat actors across the landscape have responded by shifting away from macro-based threats, even before the boom had been lowered, because they knew that the way they've been doing it was going to end. Based on Proofpoint's campaign data since October '21, threat actors have pivoted from using macro-enabled documents attached directly to messages to deliver malware, and have increasingly been using container files such as ISOs and RAR attachments and Windows Shortcut .lnk, LNK files.

So what's going on is that Microsoft blocks VBA macros based on the so-called, and I love this, the Mark of the Web, which tags files based upon their source. The Mark of the Web identifies whether a file originated from the Internet Zone within their computer. And the Mark of the Web, I just love saying that, tags the file that's downloaded. But it cannot tag files contained within other files; right? So the Mark of the Web can be bypassed by using a container file format. An IT security company Outflank - that's a great name - has detailed multiple options for red teamers to bypass - the acronym of course is MOTW, Mark of the Web, mechanisms. And of course these techniques can be used by threat actors, as well.

Threat actors can use container file formats such as, as I mentioned, ISO, RAR, ZIP, and IMG, image files, to sneak macro-enabled documents into a user's machine, bypassing that flagging. The container gets flagged, but of course its contents do not. So the file system will not identify the document as coming from the web, and some of these new defenses will be bypassed. And, you know, it occurs to me - Microsoft, hello Microsoft, are you listening? Why not mark the contents of containers? You know, you're marking the file. Mark the contents. Maybe they can't because the Mark of the Web is an NTFS property. And so you probably can't reach in. But if you watched the container being decontainerized, then you can get smarter, Microsoft. So maybe you need to get smarter.

Anyway, the cat and the mouse race continues. Having Microsoft finally hampering, thank goodness, the auto running of these macros by default was a huge leap step forward. I'm still somewhat amazed by the amount of pushback that Microsoft received. People want to have the power without being willing to take responsibility for what having it means. So Microsoft finally, after decades, did the right thing by simply taking that power away while providing them with ample safe workarounds. And again, I would argue Google is going to have to do the same thing. They're going to end up pulling the plug on third-party cookies, and people are going to scream. And the good news is they will have had years to adapt their technology. So pull the plug anyway, Google.

Okay. Lewis, who hangs out in the spinrite.dev newsgroup, he wrote sometime between last week and this week. He said: "I was surprised to hear this week that SpinRite 6.1 is still using 16-bit addressing, limiting it to 64K." And then he said, "I forgot the word you used," and he means segments, 64K segments. He says: "Is this actually helpful in 2022, and wouldn't it be simpler for the code to use 32-bit addresses rather than swap between 64K blocks?" And he finished: "Haven't we been at 32 bits for almost 40 years?"

Okay. So Lewis's question raises an interesting point about practical computing and computer science that I wanted to take a moment to address. So let's for a minute delve into some theoretical aspects of computer science which not everyone may have stopped to consider. Okay. First of all, let's understand that whatever they are being used for, the number of binary bits collected together determines the number of possible states that the collection of bits can have. We might be using a collection of bits to represent a value, in which case the number of bits determines the number of possible discrete values that can be represented by that collection. Or the collection of bits might be a cryptographic key, in which case the number of bits in the collection determines how many wrong keys that one right key is hiding among. Or the collection of bits might be a

pointer to one object in a linear array of other objects. In that case, the number of bits in the collection determines how many discrete objects that collection of bits is able to refer to.

So far, it would appear that the more bits the merrier. After all, what's wrong with being able to represent more values than you might need to? Or what's wrong with being able to refer to more objects than you could have? In all cases, logic would suggest that there will be some minimum number of bits required to do the job, but that having unused bits would not be a problem.

There are at least two problems with having unneeded and unused bits. One is power consumption. All other things being equal, and they pretty much are, 32 bits consumes four times the power of 8 bits. And those 32 bits consume equal amounts of power whether they are zeroes or ones because they still need to be checked and moved around. And speaking of moving them around, whatever the bit-size is, that is, the size of the collection, all of those bits need to be moved around. They probably need to be loaded and stored from and to main memory.

But as we know, memory bandwidth has become one of the limiting and crippling factors in modern computing. Since main memory has been unable to keep up with our data-hungry processors, all manner of multilevel caching has been deployed in an attempt to decouple the slow main memory from our super-fast, super-hungry CPUs. So if a CPU is using 32 bits to refer to a small collection of objects that could be referred to by only 8 bits, then 24 bits out of the 32 bits of precious memory bandwidth are being completely wasted. The CPU is only obtaining 25% of the system's possible memory bandwidth if, as in this example, 32 bits are being used where 8 bits would have been sufficient.

The other consequence of unused bits is the size of the program's code and data. If the code is using 32 bits where 8 would have been sufficient, the resulting code will be much larger, by as much as four times. That means it takes as much as four times more space, takes four times longer to load, and runs as much as one quarter the speed since there's just so much more code bulk for the CPU to slog through. So this all tells us that the theoretical optimal number of bits to use would be just as many bits as are required, but no more. In fact, there were, back in the early days of digital computer design, variable bit-length computers. Early serial memories such as paper tape and magnetic tape, drum and delay line, did not inherently bring along any natural boundaries. So they worked well with the idea of variable bit-length computing.

And remember that back in the '40s and '50s this was all being invented. No one knew back then what answer would be best, what the answer would be. So who knew? Perhaps variable-length computing would turn out to be the best. But when core memory was invented, it turned out to be far more convenient to "stack" the "planes" of core memory grids and to read out a fixed number of bits in parallel, one from each plane of memory. So it became natural for computers to load and store data in the same bit lengths as the core memory could produce and consume per cycle. And that approach of grouping data bits into bytes and into words having fixed lengths has stuck with us ever since.

So what does all this have to do with SpinRite? SpinRite was born when the IBM PC was using the 8088 x86 chip. It had 16-bit registers and 20 bits of addressing. The 16-bit registers could represent 64K different values. And 20 bits of addressing could refer to one megabyte of RAM. But back then, one megabyte was so much memory that it wasn't even standard. It was quite common for the first PCs to have 256K or 512K of RAM, because programs back then were small, efficient, and didn't really do very much.

The PC's BIOS, MS-DOS, and all of the DOS clones were, and still are today, 16-bit code, and they would only run 16-bit programs. Yet an incredible amount of work was accomplished by WordStar, WordPerfect, VisiCalc, Lotus 123, dBase III, FoxPro, and

many other 16-bit programs, including SpinRite. Today, SpinRite is a 16-bit program because it's still running on the 16-bit DOS operating system. And even today, I'm not unhappy about that because most of what SpinRite is doing fits nicely into 16 bits. And again, as we've just seen, more bits is not always better.

But even back then, many of those other programs were bursting at their seams within the constraints imposed by 16-bit code. SpinRite wasn't one of them. But the PC's memory was first expanded, and then it was extended using elaborate page-swapping schemes to allow data hungry tools like Lotus 123 to operate upon much larger data sets. And toward the end of all that, DOS extenders were created to host true 32-bit programs on poor old 16-bit DOS. Doing this was an act of sheer desperation. It was the definition of kludge because DOS could still only run under the CPU's real mode, and protected mode was required to go beyond 16 bits to true 32 bits. So the CPU was being dynamically switched back and forth on the fly between real mode and protected mode. It was a mess.

But SpinRite is not pure 16-bit code. It's true that it's running in real mode alongside the 16-bit BIOS and 16-bit DOS. But SpinRite 6.1, not 6.0 but 6.1, takes advantage of what must have been an inadvertent bug in the way the very first 80286 chips operated, and which bug has appeared in every chip since because Intel must have been worried about breaking backward compatibility if they were to fix this bug. The bug arises from the fact that the limitations inherent in the 8088's and 8086's real mode, which is the only mode they had, can be simulated by protected mode.

So the 80286 chips and all of the chips which have followed don't actually have real mode. They have what amounts to a "clamped down" protected mode with full access to the system's I/O hardware. So there are no restrictions there. They're basically simulating real mode, and there was always a subtle bug in that simulation. Since I've talked about this before in greater detail, I won't go into it again. But suffice to say that SpinRite uses this bug to obtain 32-bit access to any machine's first 4GB of RAM from within 16-bit real mode code. It's kind of the best of both worlds.

SpinRite itself remains small, with most of it remaining pure 16-bit code. But when and as necessary, it's able to directly address any memory that's accessible with 32-bits, which is 4GB. Doing this was necessary in order to obtain the absolute maximum possible performance from AHCI-connected drives. And boy, will it be necessary for NVMe under SpinRite 7 because it's able to read very large 32K-sector, 16MB blocks with a single command, all at once. And, boy, we've seen what that means.

But in real mode the program counter, which points to instructions, is still 16 bits long. So code must be executed within 64KB blocks. And one of those is what SpinRite has outgrown, thus now needing a second code segment block to contain its growth. Once SpinRite 6.1 is finished and published, I plan to immediately begin the work of recoding 16-bit SpinRite into true, pure, 32-bit code because it is moving to a 32-bit operating system in order to be able to boot over UEFI.

Leo: Interesting. Is it still FreeDOS?

Steve: No, because no DOS will ever be 32-bit. So it's not FreeDOS. I found a little embedded 32-bit protected-mode real-time operating system, which is like a perfect host for this. And it will be able to dual boot BIOS or UEFI. And that will just - that basically takes all the limits off of what SpinRite will be able to do. For example, in the future, running on all of a user's drives or a system's drives at once rather than one at a time.

Leo: Oh, nice, yeah. That's a great thing.

Steve: So, yeah, it'll be - it's going to be very cool. So anyway, I just sort of wanted to, you know, just sort of touch on the reality that, when I hear people especially talking about 64 bits, I just think, oh, really? Come on. I mean, again, there really are real-world problems associated with having to shuffle all those bits back and forth, if you don't need them. So okay. Some feedback from our wonderful listeners. Oh, and Leo, you forwarded this email to me this morning.

Leo: Yes. And you saw it, too, apparently, yes.

Steve: Yeah, I did. Jonathan Leitschuh - I hope I'm pronouncing your name correctly, Jonathan, L-E-I-T-S-C-H-U-H. He tweeted two tweets yesterday and, as I said, followed up with a piece of email to the TWiT gang. Anyway, I just wanted to share it. He's with the Open Source Summit. And he said: "@SGgrc and @leolaporte, I've been listening to @SecurityNow for years. This year I'm speaking at @BlackHatEvents and @Defcon, and @BSidesLV!" He said: "I learned so much from your show. I wouldn't be where I am today without it. Thank you so much for the amazing education you've provided."

And then in the next tweet he continued: "It's from your stories of great hackers like Tavis Ormandy and Dan Kaminsky and many others that I learned the norms of vulnerability disclosure. I learned the importance of building software that is secure by default."

Leo: Oh, that's nice.

Steve: So Jonathan, thank you for the tweets and congratulations on being at Black Hat and DEF CON as a speaker.

Leo: No kidding. Speaking of Dan Kaminsky, I forgot to tell you this. But Dan's mom Trudy was on the cruise with us.

Steve: Ah.

Leo: And you remember, you told quite a story about Dan's mom. She was quite famous defending her son. She told more stories like that on the cruise, including when he was 11 he'd been writing for computer magazines. And the computer publisher, not knowing he was 11, said, "I've got a big project for you." And his mom, who was very protective - Trudy was amazing. It was really nice to meet her on this cruise. She was very protective. She said, "You know he's 11?" And Dan was really mad at her, saying, "But I can do it." But yeah, she was great. And she had not heard your tribute to Dan. Of course Dan, as we know, passed away last year at the age of 42. It was very tragic. So her friend who brought her on the cruise is going to play that tribute to Dan for her.

Steve: Oh, good.

Leo: But she was, I think, really glad she came on the cruise. She'd been pretty depressed. And she came on, and she got so much adulation and got so much out of telling her stories about Dan to the group that I just thought I'd pass that along to you. It was kind of neat, yeah.

Steve: Very cool.

Leo: Thanks for joining us.

Steve: And an example of what you get when you don't expect it on one of these cruises.

Leo: Besides COVID, yeah.

Steve: Yeah. Oh. Yeah. I said what you didn't expect.

Leo: Well, yeah, I kind of expected the COVID; you're right. Did not expect Trudy. That's one of the fun things about meeting our audience. As you know, you've learned this, our audience is very diverse and sometimes very interesting. One of the guys on there was Army Human Intelligence. I said, "What do you do for the Army?" He said, "HUMINT." I said, "Oh, what's that?" He says, "Well, I do interrogations. I was at Abu Ghraib." And I went, oh. And we had some very interesting conversations. Really fascinating stuff.

Steve: Very cool.

Leo: So just some super smart, talented, interesting people. So as you know, it's always fun to go on these meetups and trips and so forth because...

Steve: Oh, and Lorrie and I had a ball when we were out traveling around meeting everybody.

Leo: Oh, people love you, yeah, of course.

Steve: So Alex Neihaus - Neihaus?

Leo: Neihaus, yeah. We know Alex well, yeah.

Steve: We know him well, a great friend of the show. The very first sponsor that this podcast ever had.

Leo: Exactly right. Thank you, Alex.

Steve: He said: "I especially enjoyed your and Leo's mirth at the astonishing flaws in the MV720 in Episode 881." He says: "I worked in China in the mid '90s. This doesn't surprise me. It's a cultural thing, a fundamental business drive to maximize profit by cheapening the product." To which I say, well, god help us.

Leo: That's what you get.

Steve: Yeah. Mark0j, he said: "Hi, Steve. Long-time listener here. Love your show. Just wanted to drop an information that there is an open source version of Tailscale that is called Headscale." That's kind of clever, head versus tail. "Seems good alternative to the paid version. Available here." And a link to it at GitHub. So Headscale is it, and I just wanted to pass that on because there is a community-supported Tailscale, but they're pushing people to subscribe, and I would also head toward the free one.

Someone tweeting as Henning said: "Dear Steve. Just listened to GRC" - oh, he said GRC-881. He means SN-881.

Leo: Close enough, yeah.

Steve: "And your criticism of the way Microsoft implemented the protection against brute force password guesses," he said, "(allow 10 failed logins, then ban any logins for 10 minutes, then allow 10 failed logins again and so on). You asked why Microsoft did not block the logins indefinitely." And frankly, once he said it, it's like, duh, of course. He says: "And this is my answer because I designed a login system for our university's web applications that works the same way. If we would block accounts, then it would be easy for anyone of our aspiring computer science students to produce a small script that locks out all of our users..."

Leo: Of course.

Steve: Uh-huh.

Leo: Oh.

Steve: In one night.

Leo: Just by attempting failed logins.

Steve: That's right.

Leo: You could lock everybody out.

Steve: Completely prone to abuse.

Leo: See, you and I - but this is a problem all security has. We didn't think evil enough.

Steve: We did not think evil; you're right.

Leo: Yeah, yeah, yeah. Wow.

Steve: Anyway, thank you very much. And to the other - a number of our listeners said, uh, Steve, you know what the problem with that is. It's like, uh, yeah. I do now. I mean, yeah, thank you. Now, many of our xkcd-following listeners sent me the following graphic. I've got it in the show notes. And it's cute. And so this is just classic xkcd. The little picture is titled "Energy tip: Increase the security of your home power supply by installing an air gap."

Leo: Yeah, now I get it.

Steve: And so it shows the cord coming in from the right to a flood lamp which is aimed at and maybe like about a foot away from a solar panel that then has a cord coming from it to a little box, presumably an inverter, that turns the DC back into AC, and off it goes. And of course I loved the concept. It's funny. But the engineer in me immediately is appalled by the inefficiencies. You have inefficiency converting the electricity to light. You have inefficiency of the light escaping in all sorts of directions. I mean, we can see the light bulb shining here from the side so that's bad. Then we have the inefficiency that's inherent in solar-to-DC conversion, and the inefficiency of a DC-to-AC inverter. So yes, I know I'm being far too pedantic. But I would not have my home air gapped for that reason.

Leo: I don't think he's serious. I think it's...

Steve: No, no, no, no. Just it's a wonderful cartoon. So thank you again, xkcd. You've provided us with many moments. Richard, somebody who's got his name as Richard COVID with five syringes and the word "Flu" in his Twitter name, he liked our mention last week and another bit of mirth from us, Leo, about this crazy Microsoft printer weird cloning copy creation. And I didn't include it in here, but he mentioned, he had a long tweet that went with this picture showing Brother MFC-J985DW Printer, and then another identical one (Copy 1). And notice it's the second one that has been set to default.

Leo: Yes, he solved it.

Steve: Yes. He noted that exactly this happened. For reasons he was never able to figure out, now he knows, the real one stopped working. The copy worked. And he said, okay, fine. He fixed it the way most people do: I'll just use that one. It's like, okay.

Leo: Oh, that's kludge-y, yeah.

Steve: Someone tweeting as Hiveware said: "@SGgrc A little pushback on your Windows OS being creaky and brittle." He says: "Take a look at 11's Settings controls. No longer are they a thin veneer over ancient code. They are a beautiful evolution of the venerable PropertySheet control, top to bottom. Credit to MS where credit is due."

Okay. So after reading this tweet and wanting to include it in the podcast for the sake of balance - I would like to be balanced - I thought I might have missed something. So I fired up a VirtualBox VM where I keep an instance of Windows 11 because, you know, it's not actually good for anything, and I went over to the Settings controls, after waiting for it to update itself somewhat endlessly. Much like Kirk on the bridge, I was unable to go to warp drive, or warp speed.

So anyway, it looks, first of all, exactly like Windows 10. Perhaps I missed what Hiveware was referring to. But I poked around.

Leo: Beautiful. What are you talking about? It's beautiful.

Steve: It's beautiful. Just like Windows 10.

Leo: Just like Windows 10.

Steve: So I poked around, and actually it was your discussion with Paul from last week, Leo, that led me to the control panel for Sound. It looks exactly like what we had, perhaps as far back as Windows 3.1.

Leo: It is in fact the old control panel. You nailed it.

Steve: Yes, it is. It is the old property sheet control.

Leo: Yeah, yeah.

Steve: Or go to Internet Options dialog. It's the same as we've had forever. Or the Device Manager, the same as we've had. And I could go on and on. So he says: "No longer are they a thin veneer over ancient code." Well, I think that's precisely what those ancient dialogs are.

Leo: Look at these. This is hysterical.

Steve: I know. Nothing has changed.

Leo: This looks like XP.

Steve: They are ancient code. And I think that's great because those are the parts of Windows that still work.

Leo: Right.

Steve: Most reliably.

Leo: In fact, when you want to change user settings, you don't want the thin veneer crap.

Steve: No.

Leo: That Microsoft has given us. You want the old school.

Steve: You don't want something that looks like a web page.

Leo: You don't want this, no. No, you don't. You don't. You want the old school. In fact, many people have learned what the command is to open those old user property settings.

Steve: Yup. So anyway, I didn't mean to diss on Hiveware, but I just, you know, I thought, okay, wait. Am I missing something here? Maybe 11 did change everything, and I just haven't dug in deep enough. But no. It's still Windows 3.1 with, I mean, it doesn't have the same borders and the same colors.

Leo: No, and it's got rounded corners now, come on.

Steve: And for what it's worth, again, I'm happy that they're leaving the stuff alone that works because for that reason, it works.

Okay. I noted Uhura on the bridge behind Kirk at the top of our show notes. And I just wanted to make a mention of her passing this last weekend. Nichelle Nichols at age 89 passed. And of course she was Uhura through all of the original series three short years, and all six feature-length films. And the bridge would not have been the same without her.

Leo: Unh-unh.

Steve: We have previous lost Spock, Leonard "Bones" McCoy, Scotty, and Nurse Chapel. Kirk and Chekov and Sulu are still with us, and we'll keep our fingers crossed that they keep going for a long time. So what an incredible legacy. There was a book, I guess I read two of them, it was called "The Enigma Cube," Enigma something. There were two of them. Anyway, there was a funny little bit there where - I'm trying to remember how - it was about time travel. And somebody, a really very gifted physicist was confessing to some friends like in the far future that he really liked this really cheesy science fiction series that he felt was really beneath him, and it only had three seasons, and it was called "Star Trek." And he was sure that nothing was ever going to come of it, but he liked it anyway. And of course the joke was that, yeah, it's still spawning new series even today, in 2023.

And also, Leo, I wanted to mention, I think we talked about it on the air last week, two series, "The Dropout" on Hulu and "WeCrashed" on Apple TV. I really enjoyed them. And for those who don't know the story of Theranos, a very young female Stanford University undergrad, who basically wanted to achieve Steve Job-style success and apparently who believed that all she needed was to want it badly enough, decided that being able to perform hundreds of blood screening tests given a single pin-prick drop of blood from a user's finger would change the world. And of course in that she was correct. That would certainly would have changed the world. But she was wrong in her belief that all she needed to do was wanting it badly enough. And she had, believe it or not, no education or training in the science that would be needed to realize her dream. None at all. There was never any reason to believe that it was possible, and there was a lot of reason to think that it wasn't. But those pesky details didn't give her any pause.

Now, being a bit of a biohacker myself and being somewhat enthralled by this presentation, I was interested in a couple things. I wanted to know how accurate the series was, and I was interested enough in the blood testing technology that this Elizabeth Holmes and her Theranos were attempting to commercialize that I followed up by watching as much of the various YouTube clips from the time and subsequently as I could, that is the real stuff, because everything's on YouTube now. And I even did some research in the U.S. Patent and Trademark Office to get some sense, as I said, for how accurate its portrayal of these events and characters was.

Leo: Well, it was based on a podcast, so you know it's accurate. "The Dropout" was based on a podcast, as was "WeCrashed," by the way. They both came from podcasts, oddly enough.

Steve: Right. So for anyone else who may be interested or curious, given the available public record, everything portrayed in "The Dropout" was astonishingly accurate.

Leo: It was well researched, yeah.

Steve: Right, oh my god, right down to the lighting used to shine off the corneas of Elizabeth Holmes' eyes...

Leo: Yes, I noticed that, yes.

Steve: ...in a series of memorable close-ups. I even found a multi-page document which dissected the "Edison" mobile lab machine that they were never able to get to work. What was clear was that, even if it had worked, it would never have been able to achieve what they were promising. In the series that we saw in "The Dropout," the original scientists who were working on the original concept ended up leaving the company through various reasons. And their work was discarded, and basically a robot glue dispenser was used to replace this thing. And it was downhill from there. But for anyone who's interested, if you haven't seen it, I really enjoyed it. I recommend it. And "WeCrashed" I thought was equally good.

You know, both are - and you and I, Leo, were talking about this before the show. I think you and I particularly, and other people in the industry, would be drawn to this because we've seen this over and over and over; right? I mean, it is the Steve Jobs personality type that is able to bring Apple back to life from the dead. I mean, remember how close

it was to being gone. And to create these things. I mean, sketchy as Elon has become, when you see those rockets come back to Earth and land, it's like, oh, god.

Leo: They work. They work, yes.

Steve: It's astonishing. And it takes people who have this kind of personality. I thought that "WeCrashed" was mostly a demonstration of that personality.

Leo: Oh, I agree, yeah.

Steve: To me it was less - yeah. And well cast. Boy.

Leo: Oh, is he good.

Steve: That guy, he was a...

Leo: Jared Leto is so good, yeah.

Steve: Oh, just fabulous.

Leo: Very believable, yeah.

Steve: So, yeah.

Leo: Yeah, I think they both are indictments of that Silicon Valley culture where you can raise a lot of money based essentially on hype.

Steve: Yes. And charisma. And the venture capitalists certainly know that for every one that succeeds big, 20 die horribly.

Leo: Right.

Steve: And burn up a lot of cash. And BMW leases are ended and blah blah blah. I mean, massive amounts of money gets burned through.

Leo: Oh, yes.

Steve: What was not clear to me about "WeCrashed" is the timeline. I think they didn't succeed in showing us that this was a 10-year span.

Leo: Right.

Steve: Because, you know, it was crammed into eight episodes.

Leo: Right.

Steve: So somehow I missed the sense of the time that went by, which I think would have, you know, it was sort of missing from that. I don't know how you do it. I mean, they kept flashing years up on the screen, the date. But they didn't convey the time that was taken. But anyway, I recommend both without hesitation, if you're interested in this idea of fundraising personalities. It's not the way I operate. When I was creating the light pen, I built one and got it to work. And then I built, like, 12, and I sold them. And I took the money from those and built a hundred, and sold them and took the money from those and, you know. Bootstrapping, the old-fashioned way.

Leo: Very old school, yeah.

Steve: Which is the way Jobs and Wozniak started in the beginning with that Apple I; right?

Leo: Right, right.

Steve: I mean, they got the Byte Shop to agree to buy some. And, who knows, they probably got a down payment from him.

Leo: They did.

Steve: And then they used that in order to build something.

Leo: There's a famous phone call Steve Jobs made to get the parts on credit, which no one would have given this hippie kid, but somehow he was a good talker, and he really - he in some ways is the prototype for both Adam Neumann and Elizabeth Holmes; right?

Steve: Yes. Well, he was clearly Holmes' inspiration. She talks about it.

Leo: Oh, yeah. Black turtlenecks and...

Steve: She went to the black turtleneck, yes.

Leo: Yeah, yeah, yeah. Good TV, too. That's, I think, the other side of that is both entertaining shows, yeah.

Steve: Yeah, really good, and really well produced. And I will close, before we talk about our show topic, with an observation that Winamp has released, is about to release, well, it's released a release candidate after four years in development. And I don't recall what caused me to mention Winamp last, but when I did, I remember like adding something, "Winamp, really?" And I was immediately scolded by a number of our listeners who said, "Hey, I'm still using it. It's the best media player ever created by man."

Leo: Yup.

Steve: And I can certainly respect that since I've created a few things that others regard as the best ever, and they've been around for quite a while. So I just wanted to give our intrepid diehard Winamp fans the news that after four years of development, the first new release candidate of Winamp has been released. Lawrence Abrams over at BleepingComputer was where I learned of this. And basically they've spent all of this time moving the code from Visual Studio 2008, which by the way is what I'm still using because it works great, to Visual Studio 2019. They now have it under 2019 so future movement will be easier.

He wrote: "Winamp ceased development after version 5.666" - ooh, that's the wrong place to stop - "was released in 2013." And he says: "That was until October 2018, when Winamp 5.8 was leaked online, and the developers decided to publish it themselves on the Winamp.com website." And I think it was that leakage online that must have been the reason I mentioned it previously because that seems about, like, 2018, about the right timeframe.

And he said: "Since then, the developers have promised an updated version with cloud streaming support and more modern features." Oh, it looks just wonderfully funky still. Lawrence has a screenshot of it, and it's so retro that it just warmed my heart, Leo. Anyway: "Finally, in November 2021," he said, "the Winamp.com website received a facelift with a new logo and a beta signup form to be notified when new versions were released.

"Last week, Winamp 5.9 RC1 Build 9999 was released, marking it as the first version released in four years and as the first release candidate of the revitalized media player. While the Winamp release candidate does not contain too many changes, the main goal of this release was to upgrade the code base from Visual Studio 2008 to 2019. Now that this has been completed, the team can add new features and capabilities to the media player." So anyway, Winamp is not dead. It hasn't been moving a lot lately, but there's still life left in it.

Leo: Not dead yet.

Steve: It's just a flesh wound.

Leo: Just a flesh wound.

Steve: That's right.

Leo: No, I think it's really cool. And it's a great product and has been for years.

Steve: The importance of language. And one of those sort of conversations and you have in college, like along with the meaning of life.

Leo: Right.

Steve: It is, okay, we think in the language that we speak. That's the way we manage our own thoughts and the way we process reality. So that means that the thoughts we have need to be expressible in the language we speak. So to what degree does our choice of language constrain what we're able to think.

Leo: Yes.

Steve: And that'll keep you up late at night. It's like, wait a minute. You mean I can't think anything I want? No. You can only think what you're able to express in your language.

Leo: This is one of the things the great Noam Chomsky, who is very important for computer science, but he was a philosopher of language. Gosh, he's 90 now, I'm seeing.

Steve: Yeah.

Leo: Wow. But great philosopher of language. And that was one of the things he was always intrigued by is how much of the language you use, how much of what you think is informed by your choice of language. And I think it's a fascinating, fascinating subject, yeah.

Steve: Yeah. Okay. Yes. Rowhammer is back, alive and kicking. Not only is Winamp not dead, but neither is Rowhammer.

Leo: Well, there you go. Nothing ever dies.

Steve: And it really shouldn't surprise any of us by now. The principle we've seen and learned over and over in context after context is that "mitigating" a security flaw is not the same as fixing it, and security flaws turn out to be stubborn things. All we need to do is look at the continuing saga being brought to us by Spectre and Meltdown, where the discovery of fundamentally exploitable flaws arising from the advanced performance optimizations which have been deeply built into modern CPUs, which can be used to leak secrets across process isolation boundaries, and we see an example of a family of flaws that similarly refuse to die.

And as we know, another completely unrelated, yet still fundamental flaw was discovered years ago, and we talked about it at the time, in the operation of today's dynamic random access memory. Security researchers discovered that modern memories - this is DRAM memories, main memories of our systems - had become so hyper-dense with every possible margin engineered out, that adjacent rows of bits which were, of course, supposed to be completely independent of one another, were in fact often able to

interfere with each other. And being the super clever engineers that they are, these researchers figured out how to turn the resulting spontaneous "bit flips" that could be induced to occur into active and effective security exploits. I mean, just the idea of leveraging a hardware anomaly like that into a security exploit, I mean, that's just - that's genius.

Of course the DRAM manufacturing industry responded, but not by reducing the storage density of RAM back down to where there would be sufficient noise immunity to prevent adjacent row interference. Oh, no. It responded with a wait for it "mitigation" of the problem by making the memory system still more complex by selectively refreshing the endangered rows of DRAM which surround the rows which might be causing interference, whether that interference was deliberate or accidental. This was named "Target Row Refresh," or TRR. And it's now built into DDR4 RAM.

So you know you're in trouble when DRAM is the subject of an apparently endless series of CVE-worthy exploits. There was the original Rowhammer. Then we had DRAMmer, which was the double Rowhammer. Then RAMBleed. Then SPOILER. Then TRRespass with TRR, so two R's, TRRespass. And then last year's Blacksmith. It's been a never-ending parade. And now, having not yet used up its nine lives, we have the less glamorously named Half-Double attack. The name for this latest attack comes - get this - from a crochet stitch which is taller than a single but shorter than a double. I know.

The point is, the actual fundamental problem of DRAM activity noise immunity being too low was never addressed. In case after case, since the cost of treating the underlying disease was too high it would have meant backing off on DRAM storage density instead of treating the disease, one symptom after the next has been treated. So today even the most recently engineered DRAM has remained diseased, with some fancy patchwork added in an attempt to shore up its fundamental problem.

I named today's podcast "Rowhammer's Nine Lives" because this problem, as I've said, refuses to die. These amazing researchers are back, having almost not surprisingly worked around the most recent iteration of Rowhammer mitigations. Their paper, which will be delivered during the same USENIX security conference next week where the TLS-Anvil will be shown, is titled "Half-Double: Hammering From the Next Row Over."

The abstract of their paper explains. They said: "Rowhammer is a vulnerability in modern DRAM where repeated accesses to one row, the aggressor, give off electrical disturbance whose cumulative effect flips the bits of an adjacent row, the victim. Consequently, Rowhammer defenses presuppose the adjacency of aggressor-victim pairs, including those in Low Power DDR4 and DDR4, most notably TRR." That's the Target Row Refresh.

"In this paper we present Half-Double, an escalation of Rowhammer to rows beyond immediate neighbors. Using Half-Double, we induce errors in a victim by combining many accesses to a distance-2 row with just a few to a distance-1 row. Our experiments show that the cumulative effect of these leads to a sufficient electrical disturbance in the victim row, inducing bit flips. We demonstrate the practical relevance of Half-Double in a proof-of-concept attack on a fully up-to-date system. We use side channels, a new technique [they invented] called blind-hammering, a new spraying technique, and a new Spectre attack in our end-to-end Half-Double attack. On recent Chromebooks with Error Correct Code (ECC) and TRR-protected Low Power DDR4 memory, the attack takes less than 45 minutes on average." And as I mentioned, they get root.

So what these guys have figured out is sheer brilliance. They came up with a way of turning the Target Row Refresh Rowhammer mitigation, which is now present in all of the latest DDR4 DRAMs, against itself. They've figured out how to use TRR to induce the problem that it's designed to prevent. Here's how they described what they accomplished.

They said: "Rowhammer is a widespread DRAM issue caused by the unintended coupling between its constituent rows. By repeatedly accessing one row, the aggressor, an attacker can corrupt data in adjacent rows, the victims, by accelerating their charge leakage. As a powerful means of bypassing hardware and software memory protection, Rowhammer has been used as the basis for many different attacks.

"Previously, Rowhammer was understood to operate at a distance of one row. An aggressor could only flip bits in its two immediate neighbors, one on each side. This makes intuitive sense. As a coupling phenomenon, the Rowhammer effect should be the strongest at closest proximity. Indeed, this assumption underpins many countermeasures which have been proposed against Rowhammer, especially the ones that rely on detecting aggressors and refreshing the charge in their intended victims. In fact, Target Row Refresh (TRR), a productionized countermeasure widely deployed as part of Low Power DDR4 or standard DDR4 chips, falls into this detect-and-refresh category.

"In this paper, we present Half-Double, a new escalation of Rowhammer where we show its effect to extend beyond just the immediate neighbors. Using Half-Double, we're able to flip bits in the victim by combining many accesses to a far aggressor, at a distance of two, with just a few to a near aggressor, at a distance of one. Both aggressor distances are necessary. Accessing just the far aggressor does not flip bits in the row that's two away, whereas accessing just the near aggressor devolves into a classic attack that's easily, and is, mitigated. Based on our experiments" - get a load of this - "the near aggressor appears to act as a bridge, transporting the Rowhammer effect of the far aggressor onto the victim. Concerningly, TRR actually facilitates Half-Double through its mitigative refreshes, turning their recipient row into the near aggressor that co-conspires with the far one that necessitates the refresh in the first place." It's just brilliant. "In effect," they wrote, "the cure becomes the disease."

They said: "While the discovery and evaluation of Half-Double is the main contribution of this work, we also demonstrate its practical relevance in a proof-of-concept exploit. However" - and boy, did they have to work to do this - "current systems limit the attacker's control, introducing four challenges," all of which they had to overcome.

"First, the adversary needs to allocate memory contiguous in a DRAM bank. However, without access to physical addresses and huge pages, we have to introduce a novel approach combining buddy allocator information with a DRAM timing side channel to reliably detect contiguous memory.

"Second challenge, ECC-protected memory can make bit flips unobservable depending on the victim data which the attacker does not control. The adversary cannot template the memory like in previous Rowhammer attacks, since hammering requires knowledge of the cell data. As the state of the art does not solve this problem, we invented and introduced a novel technique called Blind-Hammering to induce bit flips despite the error correction (ECC) mechanism of LPDDR4.

"Third challenge to overcome: Reduced address space sizes on recent ARM-based systems break the page table spraying mechanism from previous attacks. Therefore, we had to develop a new spraying technique that is still unmitigated.

"And fourth challenge: Without templating, we need an oracle telling whether Rowhammer induced an exploitable bit flip without crashing the exploit. For this, we invent and introduce a novel approach using a Spectre-based oracle for exploitable bit flips."

And here it is. They said: "We combine all of these techniques into an end-to-end successful proof of concept, the Half-Double Attack, which escalates an unprivileged attacker to arbitrary system memory read and write access. In other words, full kernel

privileges. The Half-Double Attack runs within 45 minutes on a fully updated Chromebook with TRR-protected Low Power DDR4 memory."

And they finish: "To summarize, we make the following contributions. First, we discover a new Rowhammer effect, Half-Double, and evaluate a set of devices and modules for susceptibility. We perform a thorough root-cause analysis to empirically prove that TRR is responsible for the Half-Double effect. We analyze the stop-gap mitigations present in today's systems and show that, with a new exploit using Half-Double, we can bypass them and build an end-to-end attack. And our end-to-end Half-Double Attack runs on up-to-date Chromebooks and combines the Half-Double effect with exploitable techniques, side channels, and a Spectre attack."

So the moral of our story is responding to fundamental exploitable design flaws with mere mitigations only delays the inevitable. I expect that DDR5 DRAM will learn the lessons of the Half-Double attack to render it, too, less effective by further complicating our DRAM. Maybe that will be the end of it, or maybe DRAM and Rowhammer will turn out to have still more life left in it.

Leo: Do you maybe, would you say maybe that you should eradicate, not mitigate? I don't know if that makes sense. It rhymes, though.

Steve: Hey, Leo, great marketing term. You could raise some money with that. Eradicate, don't mitigate.

Leo: Don't mitigate, eradicate.

Steve: That's right.

Leo: I'm good at writing ad copy. That's my one and only skill. That's it.

Steve: That would do it. That would do it.

Leo: Steve?

Steve: So anyway, Rowhammer's still with us. And, boy, so clever to be able to use the previous hardware mitigation in order to facilitate the next-generation attack.

Leo: Gumby says, "Leo, leave the sloganeering to the pros." Okay. Steve Gibson, you did it again, as you do every Tuesday on Security Now!. Thank you so much for being here. This is of course the one and only show you have to listen to every week just to keep up with the fast-paced world of bad guys. Steve is at GRC.com. That's the Gibson Research Corporation. There you'll find his bread and butter, his life's work, the best mass storage maintenance and recovery utility in the world, SpinRite. 6.0 currently. Somebody in the Discord was saying when is 6.1 coming out? And I told them, when it's done. Not one minute sooner. Or later. So you'll find, right now if buy 6.0, you'll get 6.1 when it comes out as a free upgrade. You can also participate in this development. It is awfully close now, I have a feeling, I feel like. But you're right not to say it. Don't...

Steve: Can't. Can't.

Leo: No, no, no. It's tempting, I know, but no. If you're at GRC.com, you'll note there is also a place where you can get, besides all the other great stuff, this podcast. Steve has 16Kb audio versions. That's unique to him. And he also has the transcripts, written by Elaine Farris. Those are very, very useful if you like to read along while you listen, but also if you want to search for something. You can search the transcripts and go right to that part of the show. Also 64Kb audio, as well. That's at GRC.com. Leave feedback for Steve there at GRC.com/feedback, or on his Twitter. He's @SGgrc on Twitter. His DMs are open. It's worth following him to keep up on the latest.

We also have a copy of the show at our website, TWiT.tv/sn, audio and video available on our site. Or subscribe in your favorite podcast client. You'll get it automatically the minute it's available. There's even a YouTube channel. If you hear something you say "I've got to send this to the boss or to the IT department," the easiest way to probably send a clip is to go to the YouTube channel, and then you can just snip out that little part and send a link to them.

If you are a member of Club TWiT, of course, you could add free versions of the shows. You also get the fantastic Club TWiT Discord, where the party goes on seven days a week. You also have the TWiT+ feed. Coming up, some exciting events. August 18th we've got Alex Lindsay's Ask Me Anything. His AMA's at 9:00 a.m. Pacific. On the 25th, the following week, Stacey Higginbotham, Ant Pruitt, and I will do the Stacey's Book Club, "Klara and the Sun." If you haven't read it yet, you've got a couple weeks, three weeks. Hurry up. Read it now. It's a great book. So if you want to join the club, go to TWiT.tv/clubtwit.