Cascading Bloom Filters

Description: CrowdStrike's president appears in person to accept the "Most Epic Fail" award. A secret backdoor discovered in Chinese-made RFID access key cards. Counterfeit and poorly functioning Cisco brand networking gear in use by major institutions, government, and military. A startling SSD performance improvement thanks to SpinRite. When is "Bing" actually "Edge," and other errata. Another useful National Public Data breach check service. And what are "Cascading Bloom Filters," and why do they offer the promise of 100% browser local and instantaneous certificate revocation detection?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-989.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-989-lq.mp3

SHOW TEASE: It's time for Security Now!. Guess who won the "Most Epic Fail" award. We'll talk about that. A backdoor found in RFID access key cards that are used almost everywhere. And then this is a propeller hat episode. We're going to have a lot of fun. Steve is going to explain a really clever technique that may solve this certificate revocation problem. We're going to study Cascading Bloom Filters. And it's fascinating. But that's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 989, recorded Tuesday, August 27th, 2024: Cascading Bloom Filters.

It's time for Security Now!, your favorite show every week, every Tuesday. We wait all week for this moment to talk and listen to Mr. Steven "Tiberius" Gibson. There he is. Hello, Steve.

Steve Gibson: Oh, Leo.

Leo: Uh-oh.

Steve: I've taken a deep breath because, boy, do we have a brain-melting podcast today.

Leo: Oh. Oh, no.

Steve: But in a good way. We've also referred to them previously as our propeller beanie podcasts, you know, the deep dive podcasts. And actually it's one of the things that our listeners keep saying that they really like.

Leo: Oh, yeah.

Steve: As much or more than anything else.

Leo: Absolutely.

Steve: The problem is there just isn't always something like that to talk about. But we have something today for our Security Now! Episode 989 for this last podcast of August - where did that month go? - titled "Cascading Bloom Filters." And although this is going to be - this is like one of those podcasts where, if you are operating heavy equipment, keep your focus on the heavy equipment because that's more important. But then listen to the podcast some other time because no one will be able to split their attention and come away feeling like, wow, I understood that.

And I don't mean to overplay its complexity. It's not. But it's brand new. It's pure, beautiful intellectual computer science. And I know that a bunch of our listeners are going to get a big kick out of it. But we have other stuff to talk about first. Although I should say that our overall page count is down because I'm quite sure that our listeners' brains are going to saturate before, you know, by the time we get to page 16. So I don't think anybody will find anything missing here.

CrowdStrike's president appeared in person to accept the "Most Epic Fail" award. We'll talk about that. Also, this is something that I've wanted to talk about for a couple weeks, and I just didn't have a chance to get it into the last couple podcasts. But it just - it upsets my stomach because a Chinese backdoor was discovered in Chinese-made RFID access key cards. And what this portends about our supply chain is really worrisome. We've got counterfeit and poorly functioning Cisco brand networking gear that were being sold to pretty much the who's who.

I wanted to share a startling customer bit of feedback about his SSD performance improvement, and I've got screenshots which he took and sent. Also a little bit of errata, when is "Bing" actually "Edge," and another couple things. Also another National Public Data breach check service which is different than the one we talked about last week, and in this instance more is better. And then, for the second entire half of the podcast, we're going to dig into - we're going to answer the question, what are "Cascading Bloom Filters," and why am I talking about them? Well...

Leo: Has nothing to do with gardening, I'm assuming. Yes?

Steve: Nothing to do with gardening, no.

Leo: Okay.

Steve: Why do they offer the promise of 100% browser local and instantaneous certificate revocation detection?

Leo: Wow.

Steve: This is what Mozilla has been working on for four years. And if people pay attention, everyone will be able to go "I understand about Cascading Bloom Filters." And boy, will you be the hit at the next cocktail party.

Leo: It's a great way, I think, to meet people, say, hey, have I told you about Cascading Bloom Filters? And then they go, oh, no, but please do.

Steve: Yeah, well, or if it's really someone obnoxious who you do not want to talk to, you can just launch into a Cascading Bloom Filter dissertation.

Leo: There you go, perfect.

Steve: And they'll say, oh, I'm sorry, I have to pee, or something. I mean, they'll come up with some - oh, is that my doorbell? I think it's my doorbell. Hold on. Yeah.

Leo: We asked AI Leo in our Discord what Cascading Bloom Filters mean. And I don't know if his answer is right because I don't know what it is. But according to AI Leo there's a term that can make even the most seasoned programmer's eyes glaze over faster than a doughnut at a cop convention. It's a nifty way to test whether an element is a member of a set. Is that right?

Steve: Yeah.

Leo: And cascading them is like stacking pancakes.

Steve: Okay, well...

Leo: Great for efficiency, but a bit messy if you don't manage your syrup. It was good so far. Right up to that point.

Steve: That's right. Now, is it on one of your shows that you were talking about printing pancakes? It's a pancake printer?

Leo: Yes, yes. That was a - we thought that was something that Craig Newmark owned at home...

Steve: That's right.

Leo: ...because he took so many pictures of it on his Instagram. But it turns out he just is in a lot of airport lounges.

Steve: But he does, he pre-syrups the plate that the pancakes drop onto.

Leo: His technique, yes.

Steve: So as Craig always does, he's an innovator. He's leading the pack. And I should mention that, oh, Leo, we have a really good Picture of the Week. This one, this one is going to be fun.

Leo: Well, I haven't seen it, but I do have it all queued up for you.

Steve: Everybody should be ready for a lot of fun for the next hour and a half or so.

Leo: Steve, I have the Picture of the Week for you.

Steve: And we're ready for your candid first look, Leo.

Leo: Okay, that's a candid first laugh because I wasn't on camera. But absolutely this is a winner. Let me pull it up for you.

Steve: Okay. So we're aware of the, like, the bragging signs that factory floors sometimes have where they say, you know, no accidents for the last 263 days; right? And they've got like little pegs where every day somebody changes the number, and they're really impressed with how they're doing. And I couldn't remember what it was, but we had something similar once where it was a smoke detector stuck on the ceiling, and it said, like, "change by" or something, or "installed by," or something that you were supposed to put the date in there, and something put something completely non-sequitur in there instead.

Anyway, so here we have, thanks to one of our listeners, a wonderful - some signage. It says: "This work center has been accident-free since," and then there's a big white field where someone's meant to be, like, putting in the date; right? Like it's been accident free since some date. And instead it says "This work center has been accident free since Joe left."

Leo: Poor Joe. Poor Joe.

Steve: So, yeah. And I did get a listener, I've already sent out 8,747 pieces of mail with the summary of the show, this Picture of the Week, the link to the show notes and so forth to those listeners of ours who have signed up to receive that every Tuesday morning. Someone wrote back, and he said, "Is that on the White House?" Anyway...

Leo: Oh, that Joe.

Steve: Joe has not left the building yet, at least not in body. So anyway, just I love that, "Since Joe Left." Right. Okay. So PC Magazine's piece was published on August 12th, to set the time, which was two weeks ago yesterday. But I felt that it was important enough to share, and we've had our hands so full with over-the-top, two-plus-hour podcasts the last few weeks. And Elaine always tells me, she says, "Oh, this is going to take a while to transcribe." It's like, "Okay, Elaine, no problem, take your time." Anyway, I just didn't have a chance to get to this. But I wanted to share it because I thought it was very cool.

PC Mag wrote: "CrowdStrike's reputation has taken a beating after last month's massive IT outage. But rather than duck the ridicule, the cybersecurity company decided to own the mistake this past weekend" - so that was two weeks ago - "by accepting a cybersecurity humiliation award for Most Epic Fail. At the DEFCON hacking show, the annual Pwnie Awards recognize achievements and major blunders in the cybersecurity industry. Past 'winners'" - and it has that in air quotes - "of the Most Epic Fail include Microsoft, the TSA, and Twitter. This year, there was no question that CrowdStrike would receive the notorious title after the company accidentally distributed a faulty security update that bricked millions of Windows PCs and servers.

"Although CrowdStrike could have easily ignored the award, CrowdStrike's President Michael Sentonas accepted it in person, which elicited applause from an audience made up of other cybersecurity professionals. Sentonas said it was important he accept the award so that CrowdStrike could own its mistakes. He said: 'Definitely not the award to be proud of receiving. I think the team was surprised when I said straight away that I would come and get it because we got this horribly wrong. We've said that a number of different times. And it's super important to own it.' He said he will prominently display the trophy'" - wonder what the trophy looks like, you know, like a garbage can? Or who knows?

Anyway, "he will prominently display the trophy he received at CrowdStrike's headquarters in Texas 'because I want every CrowdStriker who comes to work to see it. Our goal is to protect people, and we got this wrong. I want to make sure everyone understands these things cannot happen, and that's what this community is about.'"

Anyway, so the article finishes, saying: "The gesture received praise from other cybersecurity workers since it's rare for a company to accept the Most Epic Fail award from the Pwnies. Still, CrowdStrike faces a long road to repair its reputation. A pair of class-action lawsuits have already been filed against the company, demanding it pay damages for causing last month's outage. In addition, Delta Air Lines, which was forced to cancel thousands of flights due to the disruption, is also considering a lawsuit against CrowdStrike and Microsoft." Which, you know, we talked about before.

Anyway, I just thought it was so cool that CrowdStrike's president went to the trouble and had the class to show up in person, and also that the attendees of DEFCON had the good sense to applaud his in-person appearance. So, you know, yeah. Lots of damage caused. And, you know, him doing this in no way influences lawsuits that are pending and class actions and all the rigmarole and hubbub that will result. But still, you know, as a gesture of, okay, you know, yeah, we did this, and we're really sorry, I thought that was nice.

Okay. So talked about at the top of the show a piece of news that upset my stomach. I hit it early last week. But again, the podcast, last week's podcast was already overflowing, so I didn't have a chance to share it. But I think everyone needs to hear this. The news has had some coverage in the tech press, but Catalin Cimpanu's Risky Business newsletter had the most comprehensive coverage I've seen.

So here's what Catalin wrote. He said: "A security researcher has discovered secret hardware backdoors in RFID key cards manufactured by a major Chinese company. The

backdoors can allow threat actors to clone affected smart cards within minutes" - actually quicker than that - "and access secure areas. They impact smart cards manufactured by Chinese company Shanghai Fudan Microelectronics that were built using MIFARE Classic chips from NXP." And you know NXP is Philips' new name. "The chips have been on the market since 1994 and have been widely used over the past decades to create smart key cards and access badges for hotels, banks, government buildings, factories, and many other places."

Catalin then has a snippet in his reporting of a tweet from Rob Joyce, who notes: "For those who don't know, these are the RF keycards that are used in the electronic locks for hotels and other businesses. This backdoor allows trivial reading and duplication of the keycard for those that had the backdoor key." So Catalin continues: "The chips have been on the market since 1994 and have been widely used over the past decades to create these smart key cards," you know, again, hotels, banks, government buildings, factories, and even the military has these.

"Because they've been on the market for so long, they've not escaped the prying eyes of the cybersecurity community, which has previously found several ways to break their encryption and clone MIRAGE-based cards with attacks named Darkside, Nested Authentication, Hardnested, and Static Nested attacks." You get the sense that there's something about nesting in the weakness of these cards.

"Over the years, vendors have developed improved versions of their smart cards that shipped with various improvements designed to boost security and prevent some of the discovered attacks. Two of the most popular" - and these are the upgraded card designs - "are the FM11RF08 and the FM11RF08S, where the 'S' stands for 'Security improved version.' But in a paper published last week" - and so this would be a few weeks ago from now - "Quarkslab's Philippe Teuwen says that while researching FM11RF08S cards" - the improved ones - "he found what proved to be a secret backdoor baked inside Fudan cards. He discovered the backdoor while fuzzing the card's command sets and noticed that the card was answering to undocumented instructions within a specific range."

"He said: 'Apparently, all FM11RF08S implement a backdoor authentication command with a unique key for the entire production. And we broke it.' For the Fudan FM11RF08S cards, that key was" - it's a hex code - "A396EFA4E24F." That's the backdoor that opens up any of those cards. "Then, taking his research further, he also found a similar universal backdoor authentication key for its predecessor, the FM11RF08 cards. That key is A31667A8CEC1. This one also impacted many other Fudan card models, and then we get a list of numbers. It also impacted card models from Infineon and Philips, now NXP, suggesting that these companies had likely licensed their card technology from the Chinese company.

"According to Teuwen" - the researcher who found this - "the backdoor seems to go as far back as 2007, meaning that many access key cards distributed over the past 17 years can be cloned with brief physical access within seconds. As it was pointed out above, these RF access cards are based on MIFARE Classic, which were already considered insecure, but that was due to design errors and not a backdoor. Attackers would still have to spend minutes and physical resources to crack and dump a card's data in order to clone access keys configured to it. But the backdoor makes that possible, and it adds a whole new threat matrix cell."

So what I find upsetting about this is that this is an example of the true danger we face when we're outsourcing and importing proprietary microelectronics that cannot be readily audited. And this is an example, I believe, that's crucially important because this is not just a theoretical attack; right? You know, a theoretical attack is bad enough, like the fact that it would be possible to install extra chips on a motherboard to establish a secret backdoor for a theoretical attacker. But there's nothing theoretical about what was just

found. This backdoor capability was actually secretly installed into countless supposedly secure and supposedly even more secure, like, you know, their successor enhanced security RFID access cards. So this wasn't some mistake. This was deliberate.

And we're never going to know the deeper back story here. We'll never know exactly why this was done, from how high up in Chinese industry or government the decision was made to subvert an extremely popular family of authentication chip technology, and why. And it doesn't really matter. What matters is that someone said "We'll make super secure chips for you super cheap," and they were trusted not to design-in backdoor access. Which they did design in.

So we can see how Apple is being so - and why Apple is being so incredibly circumspect about the servers that they'll be installing into their data centers. We were talking about this after their announcement. At the time, that seemed like some really cool over-the-top security precautions. Now it seems not only prescient, but quite necessary. We really do have a problem with supply-chain security. And the only way it's ever going to get better will be when some means is made to thoroughly audit the design and operation of the technologies we use. You know, the problem is that the extreme benefit of the level of integration that we have today buries literally the circuit diagram, the logic diagram of this technology under layers of silicon and masks and etching and everything that goes into building chips.

Older chips, and we've seen this before, it's possible to pop the lid off the chip and take a photomicrograph of it, and semiconductor designers are able to understand from the horizontal and vertical stripes running back and forth and, like, oh, look, that's a transistor, and that transistor's connected to this one, and they can unravel that back into an actual circuit diagram. We've seen it done with, like, the 6502 and the 8088 and early generation ICs. But now that they've gotten to be multilayer and in some cases deliberately obfuscated, it's possible to hide this stuff, it's just not feasible anymore.

So I don't know what we do. But, you know, what we've seen happening over the 19 years so far of this podcast is all of the use of the Internet and our technologies moving rather rapidly from, you know, cat videos on the Internet to being utterly dependent upon this networking technology for financial and military and governmental security. And, you know, it's coming up wanting, unfortunately.

And on a related topic, we have BleepingComputer recently carrying a piece with the headline: "CEO charged with sale of counterfeit Cisco devices to government and health orgs." They wrote: "Onur Aksoy, the CEO of a group of dozens of companies, was indicted for allegedly selling more than $100 million worth of counterfeit Cisco network equipment to customers worldwide, including health, military, and government organizations.

"According to the criminal complaint, the 38-year-old Florida man ran a massive operation between at least as early as 2013 through 2022, importing tens of thousands of modified low-quality networking devices for as much as 95 to 98% off of Cisco's MSRP for the same devices, the devices these were purported to be, from Hong Kong and Chinese counterfeiters through a network of at least 19 firms in New Jersey and Florida.

"The indictment alleges these devices were sold as new and genuine Cisco products through dozens of Amazon and eBay storefronts to customers across the United States and overseas, some ending up on the networks of hospitals, schools, government, and military organizations. The fraudulent Cisco devices sold by Pro Network Entities came with performance, functionality, and safety issues that led to failures and malfunctions which, in turn, generated significant damages to customers and operations and networks. This happened because the counterfeiters who sold the fraudulent Cisco equipment to

Aksoy were modifying older, lower-model products, some previously owned, to make them look like genuine models of new and more expensive Cisco devices.

"A U.S. Department of Justice press release reads: 'As alleged, the Chinese counterfeiters often added pirated Cisco software and unauthorized, low-quality, or unreliable components - including components to circumvent technological measures added by Cisco to the software to check for software license compliance and to authenticate the hardware. Finally, to make the devices appear new, genuine, high-quality, and factory-sealed by Cisco, the Chinese counterfeiters allegedly added counterfeited Cisco labels, stickers, boxes, documentation, packaging, and other materials.'" In other words, total front-to-back, soup-to-nuts counterfeiting of the entire unboxing and receiving experience.

BleepingComputer said: "Aksoy's companies, collectively known as Pro Network Entities, generated more than $100 million in revenue, with millions lining the defendant's pockets. However, despite his efforts to fly under the radar by using fake delivery addresses, submitting forged paperwork, and breaking shipments into smaller parcels, between 2014 and 2022, Customs and Border Protection agents seized roughly 180 loads of counterfeit Cisco equipment shipped to the Pro Network Entities by co-conspirators in China and Hong Kong.

"In July of 2021, law enforcement agents seized 1,156 counterfeit Cisco devices worth over $7 million after executing a search warrant at Aksoy's warehouse. To top it all off, DOJ says that 'between 2014 and 2019, Cisco sent seven letters to Aksoy asking him to please cease and desist his trafficking of counterfeit goods.' Aksoy allegedly responded to at least two of Cisco's cease-and-desist letters 'by causing his attorney to provide Cisco with forged documents.'" So, you know, he was trying to play the game, too. And, you know, Cisco would have seen this equipment on sale on Amazon and eBay and probably bought some of it and opened it up. And, you know, there was a squirrel in a cage inside instead of an actual multicore processor capable of doing what their product alleged it could do. And of course this hurts Cisco reputation a lot in addition to just lost revenue.

So the defendant was arrested in Miami on June 29th two years ago, 2022, and was also charged in the District of New Jersey the same day with multiple counts of trafficking counterfeit goods and committing mail and wire fraud. If the charges stand up in court, and Aksoy gets sentenced to prove the truth of the allegations, this serves to show how easy it could be to infiltrate critical networks if a threat actor uses a similar approach, selling genuine but backdoored networking equipment. And again, this makes us think that Apple was not in any way over the top to be taking the lid off of their servers, doing high-resolution photography to identify the authenticity and provenance of every single component on their servers as the beginning of the process that they go through before they let one of these servers get plugged into their super-secure datacenter.

So it's very clear that the bad guys have figured out that we're implicitly trusting the supply chain every step of the way. And unfortunately there are a great many steps along the way. We know that security requires that every step be secure because it only takes the subversion of one step to break the security of the entire chain. And unfortunately, the way we are doing things at the moment, we're vulnerable to that.

Okay. So this is from a listener of ours who identified himself as Sandor. And he said: "Thank you for SpinRite." He said: "I have a 10-year-old Toshiba Satellite laptop. I installed a Western Digital Blue SSD just over five years ago." And actually he doesn't say it in here, but it's a 1TB, so it's a 1TB WD Blue SSD, just over five years ago. He said: "The warranty just expired. This Windows laptop was acting sluggish. I could not immediately identify the issue. So I ran down the checklist of items. Checked for updates and ran CHKDSK - no problems. Checked disk optimization - barely any fragmentation.

Checked the firmware on the SSD - it was up to date. Ran a quick SMART check - everything was okay.

"I pulled out SpinRite and ran level 1 and checked my system afterwards. The sluggishness was still there." And that's what we would expect because level 1 is just a read pass. He said: "Thinking about what to check next, I remembered Steve mentioning SpinRite's benchmarking utility. Ran the benchmark, and the issue revealed itself - the front, middle, and end read speeds were way off." Get this. 2.489, 17.224, 11.642, and those are all MB/s. So, wow. 2.489 MB/s, I think floppies do better than that. Anyway, then he said: "See BenchmarkBefore.jpg." And I've got that in the show notes.

Then he said: "Ran SpinRite at level 3." And that's what he wants. He said: "After level 3 completed, I re-ran the benchmarks, which showed how the SSD returned to its proper performance." So now remember before we had 2.489, 17.224, and 11.642. Now, 564.353, 563.285, 564.222. In other words, back to, from down at 2.5, 17, and 11 MB/s, to 564 MB/s across the board. And he said: "See BenchmarkAfter.jpg." He said: "The sluggishness I noticed is gone; the Satellite laptop is performing as I would expect for a 10-year-old system."

And I've got for anyone who's interested in the show notes he took a picture with his smartphone of the benchmarking output results before and after. The other thing interesting is that SpinRite looks at the speed in the middle of the drive as the best estimate of the drive's average performance. And it uses that to guess at, you know, to give the user a ballpark, you know, estimate of how long a SpinRite scan will take. Due to the low performance in the middle of the drive before, SpinRite was estimating that that 1TB drive would take it about 17.72 hours to scan. Of course, that's what old SpinRite 6 users were accustomed to seeing from a 1TB drive. Not so any longer. After he ran the level 3, SpinRite now estimates that it would be able to scan that 1TB SSD in 32.6 minutes. So from 17.72 hours to 32.6 minutes. And of course the entire drive's performance is restored to its original factory throughput.

So, you know, we are seeing more and more examples of this. Initially when we first discovered this we were shocked, but we know what's going on. It's that, especially at the beginning of the drive where the OS files reside, they are almost exclusively red. And this SSD problem known as "read disturb" actually causes the storage levels in the bit cells to be disturbed, meaning altered, when neighboring regions are red. So the SSDs are extremely good about trying to get that difficult-to-read data back. They do all kinds of re-thresholding in order to set different threshold points for what are basically analog storage cells. They end up getting your data back, but at a huge cost in performance, simply by rewriting all the data, which SpinRite's level 3 does. All the ones are really ones, and the zeroes are really zeroes, and the drive comes back up to its factory original speed.

Leo: Just like me.

Steve: That's right. Glad to have you back at your factory original speed, Leo.

Leo: You know what happened? It's really hot here. It's almost 100 degrees. And I think Comcast just died. And fortunately yesterday we put in Starlink. So I'm talking to you now via space.

Steve: Wait. You look perfectly synchronized and low latency.

Leo: It's amazing. Starlink is fantastic.

Steve: It works?

Leo: Yeah. Latency is really low, in some cases lower than the terrestrial Internet with Comcast. And it's enough bandwidth to do the show.

Steve: Wow.

Leo: So if you want to take a break now - you already took a break without me, I think.

Steve: I did, yeah.

Leo: As you should have.

Steve: I created both sides of a standard break, and we will do another one here at the beginning of our main content.

Leo: Okay. So continue on. I apologize for - nobody saw that I had disappeared, but I had. And it was always crazy. What was happening is it was coming and coming back. So we have failover. The Ubiquiti fails over to Starlink. So when Comcast dies, about 10 seconds later I come back, and the way this works, I don't have to be - I'm not originating the call, so you'll continue on without it. What happened was Comcast would drop, Starlink would come up, and then it would come back, Starlink would drop. It was a mess. So I just manually unplugged Comcast. And I might leave it that way. Anyway, continue on, my friend.

Steve: You know, Elon is such a challenge.

Leo: Yes.

Steve: But he does manage to induce people to create technology.

Leo: It's amazing.

Steve: He's like, what do you mean we can't have Low Earth Orbit satellites? I'm going to throw some up in the air myself.

Leo: Thousands of them up there, yeah.

Steve: Wow. Okay. So I have two pieces of errata to share. First, it's been brought to my attention that I've been referring to Microsoft's Edge web browser as "Bing."

Leo: Understandably. Completely understandably.

Steve: Which is obviously their illustrious search engine facility. And I'll confess that the two have occupied approximately the same undesirable location in my head. So, yeah, they were confused. But now that I'm aware of this glitch I will work to be more conscientious of my tendency to lump Edge in with Bing because, you know, they're both four letters, and they're both MSFT. Wow.

Okay. The second issue was obvious in retrospect, but it just didn't occur to me last week, and that was that the National Public Data breach search facility at Pentester.com requires the choice of a U.S. state. That's not something non-U.S. listeners who were likely affected by the breach - at least those in Canada and the UK - would have. So everyone was saying, hey, Gibson, I'm in Canada. What state is that in the U.S.? Oh, yeah, that's a problem.

Leo: I might mention, though, and you know him well, Paul Holder, who is a Canadian and a resident of both of our fine forums, did an interesting thing. Let me pull this up. He downloaded the data in the NPD breach.

Steve: Yup. He got both of the monster files.

Leo: Yeah. And he wrote a little utility, because he's a programmer, that went through it. And he says he can't confirm there's anything in the database other than U.S. addresses. The bad guy said it was U.S., Canada, and the U.K.

Steve: Ah, that's true.

Leo: So...

Steve: Okay. So what Paul got, and we talked about this in detail when we went through what Troy Hunt had found, Troy did see evidence of pollution of the National Public Data breach data with other lists that had been lumped in.

Leo: Look at this. Calgary, Alberta, Canada in the State of Texas. That's definitely corruption, yeah.

Steve: Yeah. Yeah, so it may well be that what Paul got was the good data from National Public Data, and that there was additional junk that was added in just to sort of salt it and make it look bulkier and more worrisome. Who knows?

Leo: Anyway, you may not have missed anything if you're not in the U.S.

Steve: Good point.

Leo: Maybe you lucked out.

Steve: Very good point. That may not be anything that they're doing. So I got some feedback. Listener Mike wrote. He said: "Hi, Steve. Just got off the Pentest site after doing a name lookup. Yup, I'm there, but nothing very new. We froze our credit at four separate sites many years ago, and nothing is showing up since before that happened. We moved to our current house 10 years ago, and that address does not show up. Thought you might like to know."

And then yesterday listener Kevin White wrote. He said: "Hi, Steve. Very longtime listener. If you count TechTV, I think you and I and Leo go back before SecurityNow existed!" And yes, that would be, because it was during a Call for Help episode...

Leo: In Canada.

Steve: ...that you and I were filming in Toronto...

Leo: Toronto, yeah.

Steve: ...that you said, "What would you think about doing a weekly podcast on security?" And I said, "A what cast?"

Leo: Smartest move I ever made, Steve. I've just got to say.

Steve: No, me, too.

Leo: It was a good move.

Steve: This has been really good, Leo.

Leo: Yeah.

Steve: It still is. So he said: "Last week you gave us a link to the Pentester site's NPD breach check web page. I just wanted to note that for those of us with a fairly common name" - oh, and he's Kevin White - "Pentester's check page doesn't always give out complete information." Now, he masked his middle initial. He said: "My name is fairly common, and Pentester's check page gave up listing similar names after FirstName, then middle initial 'B,' and then LastName," he said, "for example, 'B' being the middle initial. And my middle initial is located somewhere else.

"So initially I thought that I wasn't on the breach list. Then I checked" - now, here's the key - "the NPDBreach.com site" - different site, NPDBreach.com - "against my name and current zip code, and I confirmed I'm on the breach list." Then he said: "Sad face. The

benefit to the Pentester site is that it appears to show you the full number of entries there are for you on the breach list, whereas the NPDBreach site appears to just give you a somewhat distilled version of the results," he said, "a single entry for all breach list items with the same SS number, I think."

He said: "So it's best to check both sites, especially if you have a common name, or if you've lived in a bunch of different places." He said, you know, "checks against SSN, not by state. One colloquial data point," he said, "a friend was on the list, but it did not have any recent addresses for him, nothing in the past 15 years. But his wife was on the list with their most current address."

He finishes: "I guess what's somewhat nice about these circumstances is that the breach list appears to have some data reliability weaknesses that might make it a little more difficult for someone to hijack an updated or recent account. Thanks, and see you tomorrow." And he wrote this yesterday. Signed Kevin.

So I wanted to make sure that everyone knew about the https://npdbreach.com website. It also appears to be safe and legitimate. They allow the NPD breached data to be retrieved by name and some address info, or by providing a full Social Security number or a full phone number. However, now that might seem dangerous, but the site explains that the user's provided Social Security number or phone number are "blinded" by hashing locally on the browser. They similarly hashed that same data at their end, and then indexed their database with those hashes. That allows the user to retrieve their records with those hashes without the data that they're submitting to the site being exposed.

Leo: That's how Troy Hunt does password searches in Have I Been Pwned, same thing.

Steve: Yes. Yeah. Not that it makes that much difference anymore since, as we know...

Leo: It's all been revealed.

Steve: ...everything is now out there flapping on the Internet.

Leo: Wow.

Steve: So, wow, yeah. Oh, and I should mention that I did get a piece of email from someone, I think it was titled "The Seven Circles of TransUnion Login Hell."

Leo: Oh, boy.

Steve: Where, oh. And I actually had the occasion to need to log into my TransUnion data last week. Boy, is that site broken. Oh. I mean, it's just so broken.

Leo: I totally think it's intentional. Yeah. No, it's my opinion that all of these sites, they don't, look, they don't want to give you credit freezes. So they make it - but

they're legally required to. In fact, they're legally required to do it online. So they just make it broken.

Steve: Oh. Yeah, I mean, every trick in the book in order to get logged in. And I, lord knows, you know, I know the tricks. But for someone who doesn't it's like, they're just going to say, oh, well. Well, they'll probably just give them a phone call and hope that they can talk to somebody.

Leo: Yeah. In fact, that's what - I think it was Experian I had the same login issue. I showed it on Windows Weekly a few months ago. But then you call them. And I don't know why they - you wouldn't think they'd want a call.

Steve: No. You would think that would be - but I tell you, that's what they're - they're driving call traffic.

Leo: Yeah, yeah.

Steve: Okay. So this is from Darren in New South Wales, Australia. This was just a really nice note. And I wanted to share it for - not for the flattery that he gives to us, but for the perspective of a recent listener that I thought was valuable. He said: "Hi, Steve. Long-time follower of GRC, short-time follower of Security Now!. Early last year I was looking" - so early last year, right, 2023 - "I was looking for a new podcast when I stumbled across Security Now!. Despite being a long time GRC / Steve Gibson follower from the late '90s and visiting your website a lot, particularly in the early days of firewalls for the obligatory Shields Up! test on reinstall of Windows, for some reason I never clicked on the Security Now! link.

"When I came across the podcast last year I saw your name and thought to myself 'this guy knows what he's talking about, I might have a listen.'" He said: "It was Episode 909, 'How ESXi fell,' and I was hooked," he wrote. He said: "I spend quite a bit of time in the car, and I listen to the podcast. So I quickly caught up and was waiting for the podcast for next week when I remembered you and Leo speaking about the fact that GRC has all the previous podcasts available, and I thought, why not? So I downloaded episodes one through 100 and pressed Play."

He says: "What a different world it was back then when you could afford the time to spend a number of weeks telling us how the Internet works and how PKI works, as opposed to today when it just seems like there is a never-ending collage of security disasters, vendor mistakes, and just plain carelessness. Additionally," he said, "I really liked the episodes on NETSTAT (that was #49) and the History of Virtualization (#50)."

He said: "As Leo mentioned recently, compiling your tutorial sessions into a consolidated package would be very valuable for the beginner, which is why I wish I'd come across your podcast back in the day as I'm an RF Engineer who somehow found myself in the murky world of IT, and I am still wondering how that happened. The way you can explain a complex and sometimes abstract concept in simple and understandable terms is an enviable ability. At work, I have to frequently provide a translation service between Engineering and Operations, and I've used your examples and explanations a few times to great effect.

"So my routine is this: I will listen to episodes one to 10. Then I will listen to the current podcasts, usually two weeks' worth, given the time it's taken me to listen to 10 episodes.

Then I listen to episodes 11 through 20, then the current ones I've missed, et cetera. I'm now up to Episode 102. I'm wondering what my listening schedule will look like as your episodes get longer and longer over time. I'm thinking I may have to drop back to only five episodes from the before times before tuning back in to the current podcasts. Otherwise I'll be too out of date, and I don't want that to happen.

"I would recommend to all latecomers to Security Now! to go back to the start and revel in the glory that is Security Now!, wonder at how naive we were back then, and remember a time when a couple of gig of RAM was overkill and way too expensive. Congratulations on a fantastic podcast, a great website, and hopefully we can all be here listening to you and Leo describe the hopefully soft landing after the end of Unix time."

Leo: 2038.

Steve: "Thanks and regards, Darren from New South Wales, Australia."

Leo: That's great, Darren. Thanks, that's awesome.

Steve: So I also wanted just to share this as an example of how different our feedback from email can be from the "here's a link" messaging through Twitter. Now, don't get me wrong, I know there's a place for both. But the nature of our feedback is now completely transformed from the way it was and in a way that I think works for the podcast. And Darren's point about those older podcasts still being relevant really is true. You know, probably after we finally retire the podcast, Leo, I'll go back and pull together some tutorial retrospectives that, you know, what do they call those, "evergreens"?

Leo: Well, and until then, people can of course do it themselves at your site.

Steve: Yeah.

Leo: We also have, you know, all the episodes.

Steve: Well, and you've got the high-quality ones, which is good.

Leo: And video, for the ones that there were video for. It's a little tricky because you have to kind of go back a page at a time. But I'll give you a little tip. The URLs, if you type TWiT.tv...

Steve: Our uniform?

Leo: Yes, /sn and then a number, not with a zero, but just a number like sn1, that's the very first Security Now! episode. So you can - it'd probably be easy enough to write a scraper to take advantage of that fact. There's Security Now! 2, and so you could also do it this way, one by one. But it is, you are going back in time to the year 2005. Well, look at this one, Security Now! 4, we just discovered passwords. That's very exciting. I think they'll be useful.

Steve: [Crosstalk] password, what do you know.

Leo: Ah, here's your Personal Password Policy, Part 2. You probably talk about Password Haystacks or something.

Steve: I do, I do like alliteration; don't I.

Leo: Yes, you do. Keyboard sniffing, remember that, Bluetooth snarfing?

Steve: Ah.

Leo: And what were they using, ROT13 to encrypt the keyboards?

Steve: Ah.

Leo: Those were the days, my friend. We thought they'd never end.

Steve: Actually, it was a fixed XOR mask.

Leo: That's what it was. Excellent. For XORing. Crazy. Crazy. With a fixed mask.

Steve: And on that note, we should do our third and second-to-the-last sponsor because I'll take a break here in the middle. And everybody, you know, maybe some ice water against your forehead because your brain is going to heat up.

Leo: You want a propeller hat episode? Guess what? You're going to get one.

Steve: Yeah, your brain is going to heat up. But I tell you, this is a new concept, it will be a new concept for many people. It is so friggin' cool. So...

Leo: I'll make sure AI Leo listens.

Steve: And Leo, actually, I was thinking of you and your own coding of your problem solving annually. There's some value in this.

Leo: Oh, yeah. This is what, you know, I do every December the Advent of Code Calendar. I'm stuck on Day 22, I think, from last year, December 22nd. But every one of them is really about algorithms and using algorithms.

Steve: Yeah.

Leo: And so, yeah, I've always...

Steve: This is a new algorithm.

Leo: Good. I can't wait.

Steve: This is a very cool algorithm.

Leo: Can't wait. All right, Steve.

Steve: Okay.

Leo: Part 1 of our subject of the hour, Cascading something or other. Bloom Filters.

Steve: So as I said, yeah, at the top of the show, I am well aware that this podcast's enduring favored feature is pure and clean theoretical deep tech explaining.

Leo: Woo-hoo.

Steve: And of course, Leo, you used to introduce me here as the "Explainer-in-Chief."

Leo: Yes. I'll start that up again. I forgot about that.

Steve: Well, I don't know that we're going to be able to do that all the time.

Leo: Right.

Steve: You know, we were able to do far more of that back in the beginning, which, you know, 19 years ago, before pretty much everything had been thoroughly explained at least once, and in some cases multiple times. So, you know, but I know that this is what people want. So I'm always on the lookout for something to come along that we've not yet explained. And it's the nature of those things to become increasingly rare because by the end of like the next hour, we will have done Bloom Filters. And, you know, if the topic ever comes up, we'll just point back to Episode 989.

So this week we have as pure and clean and deep and theoretical a topic of computer science as we've ever had. Which, and I have to tell you, I'm excited about it, too, because it's just so cool. You know, and I just want to also say that I understand we have listeners who will write to tell me that, while they do very much enjoy the podcast, and pick up useful tidbits every week, they only, by their own appraisal, feel that they understand around 5%, they often say, of what is sometimes discussed here.

Leo: And by the way, me?

Steve: No. Anyway...

Leo: I'm right in there with you, kids.

Steve: I'm feeling somewhat self-conscious that today's topic is likely to further reduce that percentage.

Leo: I'm going to try really hard, Steve.

Steve: I know. Everybody's going to get it. But again, I saw an email from someone who says he listens to the podcast while swimming laps. And I'm just afraid he's going to drown because, I don't know.

Leo: Sink or swim time, kids.

Steve: For me, and I know for a bunch of our nerdy listeners, there's nothing more gratifying than the moment when a new concept is grasped and understood and becomes clear. So I predict that many of our listeners are going to find a lot of new ideas in what follows. So here it is. Today's topic is a technique in computer science known as a Bloom Filter. It was so named back in 1970 after its inventor, Burton Howard Bloom. And just to get everyone warmed up here, I'll explain, and everyone's going to get it completely by the end. But a Bloom Filter is an extremely clever and resource-efficient probabilistic data structure that's used to test whether an element is a member of a set.

Leo: Oh. And you do that a lot in computer science.

Steve: Yes.

Leo: That's a very common issue.

Steve: And is a revoked certificate a member of all certificates?

Leo: This would speed up OCSP.

Steve: It would probably be the answer to our dreams.

Leo: Well, well.

Steve: Which is where we're headed.

Leo: Nice.

Steve: Okay. So a probabilistic data structure that's used to test whether an element is a member of a set. But it's got some problems. Although false positive matches are a well-known and well-understood likelihood, so this thing can false positive, false negatives are impossible. They don't occur. So in other words, a test using a Bloom filter can return either a datum may be a member of the set, or this datum is definitely not a member of the set. Now, Bloom's invention is deliberately information lossy, which is where it obtains its efficiency. But as we'll see, this prevents members from being removed from the set. And as more members are added to a Bloom filter, the greater the probability of false positives becomes. So all of the math is understood, and it's all very cool. And so it's easy to optimize the Bloom filter for the things you want.

Okay. So why in the world have I suddenly apparently out of nowhere decided to talk about Bloom filters? Where in our recent discussions might it be useful to efficiently know whether something, some item, might be a member of a set of other items. And I just gave that away earlier, whether a website certificate might be a member of the set of all currently revoked and non-expired certificates. If we have a really large number of revoked certificates, and we now have web browsers reaching out to scores of other sites, you know, all of the certificate-authenticated TLS connections need to be checked.

So we need to have some really fast and very efficient means for instantly determining whether any of the many TLS certificates the browser receives, even just reading one page, because as we know pages now reach out to all kinds of different servers in order to construct themselves, whether any of those certificates have been revoked. And it just so happens that Burton Howard Bloom's 54-year old invention from 1970 is the best way to do that today in 2024.

One additional point is that today's podcast is not just titled "Bloom Filters." It's titled "Cascading Bloom Filters." The reason for this, as we'll see, is that the use of a cascade of successive Bloom filters elegantly completely solves the false positive problem that's inherently created by the Bloom filter's efficiency.

And as it turns out, this Bloom filter technology comes in very handy in computer science. And it's everywhere, without us really being aware of it. The servers of the massive content delivery network Akamai use Bloom filters to prevent the so-called "one hit wonders" from wasting their cache space. One hit wonders are web objects requested by users only once. After Akamai discovered that these one hit wonders were needlessly tying up nearly three-quarters of their entire caching infrastructure, they placed Bloom filters in front of their cache to detect a second request for the same object. Only if an object was requested a second time would they bother to save that in the cache.

Google's Bigtable, Apache's HBase & Cassandra, ScyllaDB and PostgreSQL all employ Bloom filters to bypass - get this - to bypass disk lookups for non-existent rows or columns, that is, the Bloom filter can definitely say something's not there. It's not so good about saying for sure it is there, that is, just a simple Bloom filter. We'll see how that's been fixed. But this is useful because quickly knowing when something is not, when something is for sure not in a database, significantly increases the system's overall performance.

Leo: Well, and you do that all the time. I do that all the time in coding because I want to know should I add this to a list? I don't want to duplicate it. Just like the caching problem. It's the same thing; right?

Steve: Right.

Leo: I don't want to duplicate it. So I'll just say is it here yet? No, it's not. Good, I'll put it in there. Perfect.

Steve: Yup, I think it's going to come in handy. And it might say it is when it's not.

Leo: Right, understand.

Steve: But that's better, you know, getting a no most of the time...

Leo: Right.

Steve: Well, actually, if you ever get a no, you know it's correct.

Leo: No is reliable. It's the yes that's not.

Steve: Correct, exactly. And we're going to see exactly why that's the case here in a second. Also Google's Chrome browser used a Bloom filter to identify potentially malicious URLs. Every URL was first checked against a local Bloom filter. And only if the filter returned a positive result was a full check then made of the URL. That's how Google handled the false positive problem. Microsoft Bing - and yes, I really do mean Bing and not Edge - uses multilevel hierarchical Bloom filters for its search index, which is known as BitFunnel.

Even Bitcoin once used Bloom filters to speed up wallet synchronization until some privacy vulnerabilities with the implementation were discovered, though it wasn't the filter's fault, it was the implementation. And the website Medium uses Bloom filters to avoid recommending articles a user has previously read. Ethereum uses Bloom filters to quickly locate logs on the Ethereum blockchain. So, you know, my point is this is a well-understood, very powerful technology that everybody's going to understand by the time we're done.

And we don't really care about Akamai, Bigtable, HBase, Cassandra, PostgreSQL, Bing, Bitcoin, Ethereum, or Medium. But we do care about Mozilla and Firefox, and about the fact that Firefox has always appeared to be leading the pack when it comes to actually protecting their users from revoked certificates. As we've all now seen, the other browsers, most notably Chrome, just pay empty lip-service to revocation. I mean, they've said they do it, and they weren't, as I showed 10 years ago. I should note, though, that Vivaldi is also a browser who in its default settings is also properly doing revocation, I'm sure using OCSP at the moment, the way Firefox does. And as we know, there's a bit of a problem with that. And actually I'm going to get to that in one second here.

So what do we know about Mozilla's plans for next-generation CRL (Certificate Revocation List) based revocation? Four and a half years ago, in January of 2020, Mozilla's blog posting was titled: "Introducing CRLite." They said: "All of the Web PKI's revocations, compressed." And they wrote: "CRLite is a technology proposed by a group of researchers at the IEEE Symposium on Security and Privacy 2017 that compresses

revocation information so effectively that 300MB of revocation data can become 1MG. It accomplishes this by combining Certificate Transparency data and Internet scan results with cascading Bloom filters..."

Leo: Well, there you have it.

Steve: "...building a data structure that is reliable, easy to verify, and easy to update." They said: "Since December" - so that would have been 2019 December - "Firefox Nightly has been shipping with CRLite, collecting telemetry on its effectiveness and speed. As can be imagined, replacing a network round-trip with local lookups makes for a substantial performance improvement." Meaning Firefox is not needing to do any querying of OCSP or a Certificate Authority's Certificate Revocation List. It's able to do it locally. They said although not all updates are currently delivered to clients, they've been doing this for four and a half years.

Okay. And as we know, since the CA/Browser Forum's members almost unanimously voted to drop use of OCSP in favor of requiring a return to the original Certificate Revocation List approach, we were wondering, how does that make any sense? Revocation Lists were no better and arguably worse in some ways.

So before we dig into what Bloom filters are and how they work, let's take a moment to understand Mozilla's position, which is driving this, on OCSP and CRLs. In that January of 2020 posting, they wrote: "The design of the Web's Public Key Infrastructure (PKI) included the idea that website certificates" - oh, I should mention something new is mentioned here that never occurred to me before. They said: "...included the idea that website certificates would be revocable to indicate that they are no longer safe to trust, perhaps because the server they were using was being decommissioned, or there had been a security incident. In practice, this has been more of an aspiration, as the imagined mechanisms showed their shortcomings." In other words, we'd like to know if certificates have been revoked, but we just can't figure out how.

So they said: "Certificate Revocation Lists quickly became large, and contained mostly irrelevant data, so web browsers stopped downloading them. The Online Certificate Status Protocol (OCSP) was unreliable, so web browsers had to assume, if it didn't work, that the website was still valid." As we talked about, you know, failing open. They said: "Since revocation is still critical for protecting users, browsers built administratively-managed, centralized revocation lists. Firefox's OneCRL, combined with Safe Browsing URL-specific warnings, provide the tools needed to handle major security incidents, but opinions differ on what to do about finer-grained revocation needs and the role of OCSP." And as we know, although they didn't mention it here, Chrome said, oh, yeah, we have CRLSets. We're all covered. It's like, yeah, but they don't work. We proved it 10 years ago.

So they said: "Much has been written on the subject of OCSP reliability; and while reliability has definitely improved in recent years," they said, "per Firefox telemetry for failure rate, it still suffers under less-than-perfect network conditions." Here it is. "Even among our beta population, which historically has above-average connectivity, over 7% of OCSP checks time out today." So 7% of OCSP you just don't get a response from whomever you were asking.

They said: "Because of this, it's impractical to require OCSP to succeed for a connection to be secure; and, in turn, an adversarial monster-in-the-middle can simply block OCSP to achieve their ends. Mozilla has been making improvements in this realm for some time, implementing OCSP Must-Staple, which was designed as a solution to this problem, while continuing to use online status checks whenever a server fails to staple a response.

We've also made Firefox bypass revocation information for short-lived certificates; however, despite improvements in automation, such short-lived certificates still make up a very small portion of the Web PKI because the majority of certificates are long-lived."

And I should just mention here that when they talk about "short-lived," they're talking about hours or days, not months. Thus they consider Let's Encrypt's 90-day certs to be long-lived, as they should because that's now the majority of the web. And what Mozilla says next is quite interesting. It's something we've never considered before.

They wrote: "The ideal in question is whether a Certificate Authority's revocation should be directly relied upon by end-users." What? They said: "There are legitimate concerns that respecting CA revocations could be a path to enabling CAs to censor websites. This would be particularly troubling in the event of increased consolidation in the Certificate Authority market. However, at present, if one Certificate Authority were to engage in censorship, the website operator could go to a different Certificate Authority."

Leo: Yeah, they would just route around it.

Steve: Yeah. And they said: "If censorship concerns do bear out, then Mozilla has the option to use its root store policy to influence the situation," is the way they put it, "in accordance with our manifesto," they wrote. So that's quite interesting. They must not be talking about typical Certificate Authorities in the U.S., but perhaps CAs located in repressive governments which might be forced to revoke the certificates of websites whose certificates had been issued by them which the government later no longer approves of.

And then I assume that since Mozilla would certainly disapprove of Certificate Authorities being used as censorship enforcement, that's where Mozilla's root store policy would come in, with them completely withdrawing all trust from any such CAs' signatures, much as the industry recently did with Entrust. And then they finish up, saying: "Legitimate revocations are either done by the issuing Certificate Authority because of a security incident or policy violation, or they're done on behalf of the certificate's owner, for their own purposes. The intention becomes codified to render the certificate unusable, perhaps due to key compromise or service provider outage, or as was done in the wake of Heartbleed.

"Choosing specific revocations to honor while refusing others dismisses the intentions of all left-behind revocation attempts. For Mozilla, it violates principle six of our manifesto, limiting participation in the Web PKI security model." In other words, as I said, they would be upset with any Certificate Authority that attempted to play games with their customers' certificates revocation status. So they kind of like being able to decide for themselves rather than always have it be the CA. And that's one place where requiring stapling of OCSP status becomes a problem. This allows the browsers to retain more control of that.

And then switching from issues of policy back to issues of technology, they said: "There is a cost to supporting all revocations. Checking OCSP slows down our first connection by around 130 milliseconds. It fails open if an adversary is in control of the web connection. And it periodically reveals to the Certificate Authority the HTTPS web host that the user is visiting." Okay. So they said: "Luckily, CRLite gives us the ability" - I guess I should just say CRLite - "gives us the ability to deliver all the revocation knowledge needed to replace OCSP, and do so quickly, compactly, and accurately."

Can CRLite Replace OCSP? Firefox Nightly users are currently only using CRLite for telemetry. But by changing the preference in some of their settings it can be entered into

"enforcing" mode. And they said there's not yet a mode to disable OCSP; there'll be more on that in subsequent posts. And that was four years ago. There certainly is, as we discussed last week, in Firefox an OCSP setting. It's enabled by default. You can turn it off. And we see the effect of it being turned off. I've not tried, I don't think I've tried turning OCSP off and then turning on CRLite, although I'm not on Firefox Nightly. And I think it's still not everybody who's got this. It's only their beta testers who are able to play with this.

Okay. So let's talk tech. We can broadly classify major cryptographic algorithms into two classes, lossy and non-lossy. All encryption is non-lossy, meaning that encrypting a source plaintext into a ciphertext does not lose any of the original's information because decryption, the reverse process, perfectly recovers the original text.

By contrast, cryptographic hashing is a deliberately lossy process. And for what a hash provides, that's what we want. As we know, a hash, also known as a digest, takes an original plaintext of any size and reduces it to a single fixed-length output consisting of some number of bits. The number of bits is chosen to be high enough so that the probability of two different texts hashing to exactly the same combination of every bit is astronomically small.

Now, common sense tells us that it would not be possible to take the entire text of "War & Peace," hash it into a tiny 256-bit - comparatively tiny - 256-bit digest, then somehow reverse that process to get the original text back. "War & Peace" is definitely not all in there in those 256 bits, even though every character of "War & Peace" has an effect upon the hash's final output.

Okay. So now let's look at the design of Bloom filters, an entirely different system which, like a hash, is similarly deliberately lossy. Any Bloom filter starts with a large array of binary bits all set to zero. The optimal size of the large array is determined by many interacting factors, but for our purpose we're going to use an array containing 1,048,576 bits. That's exactly 128 Kbytes. It actually happens to be 2^20. So that's a nice number. So basically a megabit; right? 1,048,576 bits.

Now imagine that we want to "add" an awareness of a revoked certificate to this currently empty Bloom filter, this one million bits which are all initially set to zero. We need to use some fixed characteristic of the revoked certificate. All certificates contain a thumbprint. It was traditionally generated by taking the SHA1 hash producing 160 bits of the entire certificate, and later that's been upgraded to 256. So older certificates have 20 bytes or, as I said, 160-bit thumbprints; and newer certificates have 32 bytes, or 256-bit thumbprints. But every certificate has a thumbprint.

Now, one of the cool things about a cryptographically strong hash function like SHA1 or SHA256, which we've talked about before, is that all of the bits in the hash are equal. That is, created equal. A smaller hash can be taken simply by using some subset of the whole hash's entire value, and it doesn't matter which smaller subset is chosen. So for our example, we're going to take the lowest 20 bits of the revoked certificate's thumbprint hash. Thanks to the thumbprint being a hash, these are effectively pseudorandomly set bits. Because 2^20 is that 1,048,576, the lowest 20 bits of a thumbprint can be used as the index into our bit array. In other words, it can be used to select exactly one bit from our one megabit array. And we set that chosen bit to a one.

Okay. Now let's say that we add more, many more revoked certificates to this blossoming Bloom filter. Let's say that we add a total of 100,000 revoked certificates. Since adding each certificate entails taking the lowest 20 bits of its pseudorandom but fixed thumbprint, and setting the corresponding bit in the bit array to a one, this would mean that up to 100,000 bits out of our total of one million would have been set after we added those 100,000 revoked certificates.

I say "up to 100,000," though, because there's a problem. There would be a good chance that out of those 100,000 certificates, many of the certificates could share the same lower 20 bits. No certificate is going to share its entire thumbprint, 160 bits or 256 bits. The chances of that are astronomically low. But we're deliberately taking only a piece of the whole hash, so collisions are possible. And as we know, the surprising fact is that the number of these collisions turns out to be higher than we might intuitively suspect.

We've talked about the "birthday paradox," which explains why the probability in this case of any two certificates sharing the same lower 20 bits is higher than we would think. So in practice, this means that fewer than 100,000 sets, bit settings, will have been made, changing these bits to one, due to collisions where the bit that was set by a new certificate being added had already been set to one by the addition of a previous certificate.

Okay, now, also note that after adding 100,000 certificates, less than 10% of our total bit space of one million, a little over one million, has been used. Bloom, to his credit, realized that this was inefficient. And he had a solution: Instead of only setting one single bit per certificate, let's set several. And we can do that easily by taking successive 20-bit pieces of the certificate's thumbprint and using each successive 20-bit chunk to set an additional bit. So let's set five bits in the one megabit array for every certificate we add to it.

So we start again with an empty array with all of its 1,048,576 bits set to zero. We add the first certificate by taking each of five chunks of 20 bits - remember it's got 160, so we're just going to use 100 of those. We take each of five chunks of 20 bits from its thumbprint, using each chunk to determine which bit to set. After adding that one certificate, we'll almost certainly have five bits set. It could be that a certificate might collide with itself, though the chances are slight. My point being that, since we're taking five 20-bit chunks, it's like there's a one in a million probability that two of them, actually it's lower than that, but, I mean, it's more probable than that, that some pair of them might have the same 20 bits. So it's possible.

But probably five bits are set which are representative of 100 bits of that certificate's thumbprint. And then as before, we're going to add a total of 100,000 certificates. So each certificate will be setting its five very specific bits to a one. Now we'll have somewhere less than half but close to, but less than half of the array's bits set due to collisions. But we'll be very nearly at 50%, right, because we've got 100,000 certificates. They're trying to set five bits each. So that would be 500,000 potential, which is half of our one megabyte. So about half the bits are set, a little less than half, due to collisions. And Leo, I see that you've got the cold soda can on your forehead. That's good.

Leo: So far I'm following you, though. This is good. This is good.

Steve: Yeah. I think everyone's...

Leo: We've got basically a megabit picture with five bits set for each certificate that tells us, really the only thing it tells us is there's a collision, that that certificate, if those five bits are set, is colliding with another certificate. Not that they're 100% matching.

Steve: Well, yes. So far - but so all we're doing, basically we're OR'ing ones into the bit array.

Leo: Yes, right.

Steve: And so what we have is, as you said, a picture, and but we may have collisions. We haven't yet used it, but we're going to next. Let's take our last break.

Leo: I'm loving this.

Steve: And then I'm going to show people how we use this to determine something about a certificate.

Leo: Okay. I think, I kind of think I know where you're going on this.

Steve: I know. It's cool stuff.

Leo: It's not that - it's not really complicated.

Steve: No, it's not really complicated.

Leo: Have we got to the cascading part yet?

Steve: No. And that's where you're really going to need two cans. You're going to need two cans of soda on your forehead.

Leo: Okay. All right. Let us get back to Bloom filters. And are you ready? Because it's time to cascade.

Steve: Not yet. Not yet.

Leo: Okay.

Steve: First we have to see how they actually work. So we've designed a filter where five bits get set based on five pieces of information from a certificate's thumbprint, which is the hash of the certificate. And we've added 100,000 certificates, 100,000 revoked certificates to this filter.

Leo: I understand how this works because, if those bits are set, you can pretty - if they're set you say that there's a likelihood, but not a guarantee that that certificate's in the megapixel.

Steve: Correct.

Leo: But if it's not there, you know it's not there. Right? It couldn't possibly exist.

Steve: That's exactly it.

Leo: Yes. That makes sense.

Steve: So let's say that our web browser receives a certificate it wishes to check. It wants to ask the question, has this certificate previously been added to this Bloom filter? To answer that question, we take the same five sets of 20 bits from the candidate certificate's thumbprint.

Leo: And that's a fast thing to do. It's really quick, yes.

Steve: Oh, yes. That's the other thing. All of this is super fast. We successively use each set of 20 bits to test each of those five bits in the Bloom filter. Here's what we learn from that, and think about this. As you said, Leo, exactly, if any of the five tested bits are zero...

Leo: You're done.

Steve: ...then we absolutely positively know that the certificate in question could never have been previously added to the Bloom filter's bit array.

Leo: You can see why this would be fast. In assembly language, those compares are almost instant; right?

Steve: Yes. In fact, Intel has a single instruction...

Leo: Oh, of course.

Steve: for querying a bit in a large field.

Leo: Are you set? Yeah.

Steve: So, yeah. So if any of the five bits that are tested are zero, then we absolutely positively know that the certificate in question could never have been previously added to the Bloom filter's bit array because, if it had been, then we know that all of its bits would be ones. So if any of the five bits are zero, it's not revoked. The certificate is definitely not revoked.

But the problem we still have, if all of the certificate's tested bits are ones, is that we cannot be certain that this is not a good certificate whose five chosen bits just happen to collide with bits that were set by any of the other 100,000 revoked certificates. And remember at the top I said earlier, "Although false positive matches are a well-known

and well-understood likelihood, false negatives are not." So in other words, a test using a Bloom filter can return either a "datum may be a member of the set" or "this datum is definitely not a member of the set." The "definitely not a member of the set" is what we get when at least one of the five bits are zero, since that can only happen if this certificate was never added to the filter.

Leo: This is efficient also because it cuts short the search. The minute you find a zero you're done. You don't have to keep going.

Steve: Yes, you are indeed a coder, my friend.

Leo: And if you had fewer than five bits, collisions would be more likely. If you had more than five bits, collisions would be less likely. I guess that they figured out five is kind of a nice balance.

Steve: Oh, the math is hair curling. It's got all kinds of calculus that you don't want to even think about.

Leo: You can also do it by trial and error.

Steve: But there are, given the size of the various sets...

Leo: Right.

Steve: ...it's possible to exactly zero in what you want.

Leo: What's optimum, yeah.

Steve: Okay. So what do we do about this false positive problem which is created by a Bloom filter? First, it's worth noting that many applications of Bloom filters are not sensitive to false positives. If Akamai occasionally caches a "one-hit-wonder" file because the file's Bloom bits happen to collide with another file, who cares? An extra file gets cached, big deal. Or if Medium is using individual per-user Bloom filters to avoid having users see the same article twice, but that occasionally misfires and shows them something they saw before, again, who cares? Or if a database mistakenly believes that a record exists and goes to look for it instead of immediately rejecting the request thanks to a Bloom filter failure, that's okay.

So before we solve the problem of Bloom filter false positives, I want to clearly make the point that a simple Bloom filter that is able to almost instantly make a go/no-go determination which never generates a false negative but occasionally generates false positives, can still be an extremely useful tool in computer science.

Leo: Sure. You just have to use it in the appropriate place; right?

Steve: Right.

Leo: Yeah, that makes sense.

Steve: Yes. There are, now, there are several important points, and you already touched on one, which are worth pausing to highlight here. The first is that, in return for the Bloom filter's discrimination "fuzziness," we get amazing efficiency. The hashes of 100,000 certificate thumbprints can be sorted, but they cannot be compressed because thumbprints are, by definition, incompressible pseudorandom data. Each modern SHA256 thumbprint is 32 bytes. So any simple list of those 100,000 revoked certificate thumbprints would require 3.2 megabytes just to store the list. Yet our trained Bloom filter requires only 128K. So in return for some inaccuracy, we obtain a factor of just shy of 25 times compression. So it is super efficient.

And then there's speed. As you said, Leo, searching a 100,000-entry sorted list is still time-consuming. And more sophisticated tree structures consume more space and still take time to traverse. By comparison, consider our Bloom filter. The certificate we want to test already offers its thumbprint. We need to perform five tests, each one which directly tests a single bit of the Bloom array. For those who are not coders, any coder will tell you that directly addressing and testing a bit in an array is one of the fastest things a computer can do. And as I mentioned, Intel has an instruction that does exactly that. That's all it does. You issue the instruction; you get the result instantly. So nothing could be faster than these tests.

Now consider that with the Bloom array having fewer than half of its bits set, each of these tests has a better than 50/50 chance of selecting a zero, and that the instant we hit any zero in the array when we're doing our five tests, we know that this certificate being tested cannot be represented within this array. So with five tests, each super fast, and each having a better than even chance of giving us an immediate out and go-ahead, this has got to be the fastest, most efficient and economical way of classifying a certificate as being a member of a larger set.

But unlike Akamai, Medium, and the databases that can tolerate false positives, this application of Bloom filters for certificate revocation absolutely cannot tolerate any false positives. We can never tell a user that a remote server has just served them a revoked certificate when it has not. So now what do we do? You get extra credit if you guessed this is where the "Cascade" enters the picture. Yes. To pull this off, it is necessary for the creator of the final CRLite Bloom filter cascade to have access to the entire real-time active certificate corpus. In other words, a record of every single unexpired certificate, both valid and revoked.

Leo: How big is that?

Steve: It's big. But we don't need it, just the persons building the filters need it.

Leo: Right.

Steve: Now, although this might appear to be a tall, if not impossible order, the PKI (Public Key Infrastructure) actually already has all of this in place. All Certificate Authorities are required to publish every certificate they issue into a Certificate Transparency log. That log exists, and it's available. So check this out:

A first and largest Bloom filter bit array, the one we've been talking about, is first populated by adding all known revoked certificates. As we've seen, this will generate an array with lots of one bits set. And we know that, once this has been done, any certificate that has any zero cannot be a member of the set of all revoked certificates. We also know that each of the five tests has a better than even chance of returning a zero, so most good certificates will be identified almost instantly. But for this application we must enforce a zero-tolerance policy.

Every non-revoked valid certificate in existence is then run through this first-stage Bloom filter to explicitly detect every possible false positive that the first-level Bloom filter produces. And what do you think we do with those? That's right. Those false positives are used to train the second-level filter in the Bloom filter cascade. In other words, the first filter is trained on revoked certificates. Then all the non-revoked certificates are passed through it, and any that come out because they're valid, those represent false positives created by the first level.

So those certificates are added to the second-level Bloom filter to contain all of those. So this second-level cascade, the second-level Bloom, can be thought of sort of as a whitelist to handle the false positives which are generated by the first-level Bloom filter. Thus any certificate that is not in the first level must be valid. We know that. That's the immediate okay.

Then, any certificate that was identified by the first level, which we might call a candidate revoked, and is not also in the second level, must actually be revoked since the second level was trained on valid certificates that were thought to be bad by the first level. We know that Bloom filters false positive, but they never false negative. So any certificate that the first level finds which the second level does not find must not have participated in the second level's valid certificate training. So we absolutely know that it had to have been revoked.

However, there's one last bit of mind-melting logic to cover. Since Bloom filters are known to produce false positive matches, there's a chance that the second-level Bloom filter might produce a false positive. That would lead it to believe that a revoked certificate that was found by the first-level filter and had been found to be good by the second-level filter should be trusted. So believe it or not, a third-level filter is used to catch and categorize those cases that may slip through the second-level filter. But that third-level filter is the final stage of the Bloom filter cascade, and its decision is final.

So if the first-level filter does not produce a match, we immediately know that the certificate must be valid since that first level was trained on revoked certificates, and Bloom filters never false negative. But if that first-level filter does produce a match, we then test the certificate against the second-level filter. If it does not produce a match, we absolutely know that the certificate must have been revoked, since the second-level filter was trained on valid certificates. And, again, Bloom filters never false negative.

And finally, since the third-level filter is again trained on revoked certificates, if it does not find a match, we absolutely know that the certificate is good. And if it does find a match, thanks to the pre-weeding out of the earlier filters, we can finally know with certainty that the certificate is a member of the revoked set and must not be trusted.

Leo: It's sufficient to three levels.

Steve: Yes, exactly. And I should note that since the second-level Bloom filter only needs to be trained on the valid certificates that mistakenly false positive match at the first level, that total number of certificates is very small, so much so that a much smaller

bit array and smaller filter will suffice. And that's even more true for the third level, that's only dealing with really exceptional cases. So we obtain significantly greater efficiency with successive levels of the cascade.

Leo: You don't need a megabit on level two.

Steve: Right. Nothing near that.

Leo: Maybe half that, and then half again or something like that.

Steve: And so although this, I get it, is all a bit mind-bending...

Leo: Oh, super cool, though.

Steve: It is super cool. What we now have is a way of constructing an efficient means of rapidly categorizing certificates as valid or revoked. In order to do this we need to train a trio of Bloom filters on every possible valid and revoked certificate that they might ever encounter in the wild, and that's actually not impossible. We can do it. Once this has been done, and Mozilla has been doing this now for several years, four years, we wind up with a small local certificate classifier that every user can efficiently run in their own browser. And it can instantly identify any revoked certificate from among millions of candidates without ever making a single error.

This amazing local browser capability is supported and made possible by a massive, continuous, behind-the-scenes effort to produce and distill, four times per day, those three simple Bloom filter bitmaps, which are then made available to every browser over a common content delivery network. Individual Firefox browsers reach out to obtain the latest set of bitmaps every six hours. If or when all browsers have access to these filters, we'll finally have solved the certificate revocation problem with both low-latency and total privacy, since every browser will then be empowered to make these revocation determinations locally.

Leo: So cool.

Steve: It is so cool.

Leo: And now as an exercise for the listener, why is it that you're testing revoked, keeping track of revoked certs instead of keeping track of good certs? Should be easy to figure out. It's kind of fun. I love this.

Steve: Isn't that cool?

Leo: It'll be in Advent of Code this December. He's very timely with his algorithms.

Steve: And it's such a cool technique to use in order to classify something. And, you know, the fact that most things are going to immediately reject, the top-level Bloom will immediately reject most good valid certs. Only where there are collisions do you then need to train the second one on the valid certs that are known to collide. It's just - it's elegant. And I'm impressed. I mean, to me it feels like Mozilla is out there paving the way for all the other browsers to steal it, you know, like steal the cool tech that they come up with. And then, I mean, because, you know, this is all just going to get copied by everybody else.

Leo: Yeah. As it should be. This is a great replacement for OCSP. This is it.

Steve: Yes. I mean, it is it. The problem is, as we know, Mozilla's endangered. Firefox is endangered because...

Leo: Yeah, but I can see Google adopting this pretty quickly.

Steve: Oh, yeah.

Leo: I mean, this seems like a very clever - and it's nice because all the heavy calculation, which is calculating the hashes, is done on server-side.

Steve: Exactly.

Leo: And just ports these small bitmaps.

Steve: Exactly.

Leo: There are maybe a mega, let's see, a megabit is, what, 1K? It's 128K.

Steve: Yeah.

Leo: So maybe you throw in every six hours 156K or something. It's nothing.

Steve: Right. And, yeah...

Leo: Brilliant. And instantaneous results.

Steve: Yup.

Leo: Now, I still have to understand why you only need three layers to guarantee. You're doing 20...

Steve: The math is there. And it turns out that you can show that the first filter will have resolved everything that the third one would.

Leo: Got it. So now it's a loop.

Steve: So all the mistakes are taken out, yeah.

Leo: Very interesting. I will read up on this and Professor Bloom, who thought this all out.

Steve: 1970.

Leo: Many years ago.

Steve: I was a freshman in high school, and he's like, you know, if you had a lot of bits, and you needed to get them straightened out, this is what you're going to do.

Leo: I want you to, some day when we've got some time and there's no security issues to talk about, do the Dijkstra algorithm, Edsger Dijkstra's search algorithm. I bet you could explain that pretty clearly. That's fun. Really good stuff. This is why we love Steve; right? Aren't you glad you listened to this episode? And if it's all a mish-mosh in your mind, listen again. It'll come clear. And by doing so, you're building brain cells. You're making yourself smarter.

Steve: And you're having much less room for trivia.

Leo: I have no idea who Jason Momoa is married to, or not. Doesn't - it's not in my head because I've got Bloom filters.

Steve: Yeah, like I said, you want to terminate an annoying conversation at a cocktail party, bring out the Bloom filter.

Leo: But I have to say, if the person you're talking to says, "Oh, that's interesting," you've got a keeper. Right? Now you know. Now you've got something.

Steve: That's true.

Leo: Somebody you're going to want to talk to some more.

Steve: That might be a good first date question.

Leo: No, it's not.

Steve: Maybe save that for later.

Leo: Steve Gibson's at GRC.com, the Gibson Research Corporation. His bread and butter is there, too, of course, SpinRite.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details: http://creativecommons.org/licenses/by-nc-sa/2.5/