
# Memory Hard Problems

**Description:** After catching up with a bunch of fun and interesting news of the week, Steve and Leo examine the future of anti-hacking password scrambling and storage with the introduction of "Memory Hard Problems," which are provably highly resistant to massive hardware acceleration.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-388.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-388-lq.mp3

---

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We've got a lot of security news and an acronym for Java, which I don't think is official. Also a warning. If you're about to install Java, there's something sneaky Oracle's up to. And then we'll take a look at something called "memory hard problems." Too hard for me. Maybe you'll understand. Next, on Security Now!.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 388, recorded January 23rd, 2013: Memory Hard Problems.

It's time for Security Now!, the show that protects you and your loved ones online; your privacy, too. And here's the guy who does it all for you, Mr. Explainer in Chief himself, Steve Gibson.

**Steve Gibson:** We have a massive infotainment podcast for our listeners this week, my friend.

**Leo:** Really. An infotainment podcast.

**Steve:** Yeah, I've been listening, through reading the mailbag and looking at tweets, people just enjoy the podcast. I mean, they appreciate learning something, but they're also having fun. And you I think help bring the fun to it, which is a good thing.

**Leo:** It's news you can use. Actually some viewer brought this fun to it. I showed you this before the show. And it is apropos today: "Keep calm, and disable Java on your browser." It's a play on the Keep Calm signs from Great Britain during the

Battle of Britain in World War II. What did it say, Keep Calm and...

**Steve:** Carry On.

**Leo:** Carry On. In this case. And it's instead of the royal crown it has the head of Steve at the top there.

**Steve:** Looking down with some disdain. You're running Java in your browser? What's wrong with you?

**Leo:** It's apropos because I guess there's another, god, I can't believe it, hole in...

**Steve:** Yes, and in fact someone tweeted a great acronym that Java stands for, so we're going to briefly delve into acronyms we love. I did a little research and found, oh, my goodness, some - my favorite from the old days, because I was actually a - I was a Fiat owner twice. The little X19 Fiat...

**Leo:** Fix It Again Tony.

**Steve:** I love that one. Or, wait a minute, oh, Feeble Italian Attempt at Transportation.

**Leo:** Oh, yeah. It depends on what kind of car you have because there's also - Ford also has Found On Road Dead, I've heard of that.

**Steve:** Fix Or Repair Daily, that was always my favorite Ford one, yeah.

**Leo:** Well, we'll have a few acronyms for you. And we're going to talk about, now, is this a typo, "memory hard problems"?

**Steve:** That's what the crypto industry has named it. And I think, when people understand what memory hard problems are, they'll go, okay, yeah, I can remember that. That terminology makes sense.

**Leo:** Okay.

**Steve:** To give a little tease for what we'll be talking about after we catch up on all of the week's craziness, we're seeing clearly, we're talking about it always, people are tweeting me constantly, the fact that people are building machines full of GPUs, graphics processing units, in order to create these massive password-cracking machines. They're also building them to mint bitcoins because bitcoins are a hashing problem. Now, the problem with hashing is that one of the criteria for selecting hashes has always been that

they be fast because we don't want to burden computers with a lot of work to create a hash. And, for example, one of the reasons that the AES cipher, Rijndael, was chosen is that it lent itself, the algorithm lent itself to easy implementation, both in silicon and in software.

So while all of those are good properties for ciphers and hashes to have, that's exactly what you don't want if you're using those to protect passwords. And that's why we keep - now what the industry has done is it's said, okay, we're not going to hash the password once. We're going to hash it a thousand times. Or 5,000 times. Or something. And the problem is that you can't really solve the problem that way. That is, that's a linear scaling of difficulty, and there are problems with that. So we're going to talk about the problems with that, why just things like - we've talked about password-based key derivation, that PBKDF2, which everyone is using. But we're heading toward a new solution which solves the problem in a way that is much more robust, and that's the domain of memory hard problems.

**Leo:** Got it. I guess. Sort of.

**Steve:** So a great topic for the week. And, oh, all kinds of crazy fun stuff, too.

**Leo:** Can't wait. Security Now! is on the air. Get ready. Put your beanies on, your propeller heads. Did you see, by the way, that two bitcoin-based casinos did their annual reports, and they made a lot of money?

**Steve:** I didn't know there were such things.

**Leo:** I didn't either. But this is in Ars Technica today. SatoshiDice and bitZino. They don't take money because in the U.S. it's illegal to do that.

**Steve:** Wait, Satoshi was the name of the developer of Bitcoin. So that's probably where that name came from.

**Leo:** Maybe Satoshi decided to cash in. They did apparently a half a million dollars in business. I don't know, anyway. But is it money or what? SatoshiDice is based in Ireland. It's a pseudorandom number generator game. You choose a number, then bet on the likelihood that the rolled number is greater than the one they've selected. It sounds like keno a little. But I don't know what it is.

**Steve:** Oh, interesting.

**Leo:** House has a 1.9 percent edge. That's the profit. Day in, day out. That's the beauty of these things.

**Steve:** Wow.

**Leo:** It's just math. Players put down 2.3 million bets in May through December 2012.

**Steve:** And so betting is with bitcoins? You bet bitcoins and you are paid in bitcoins?

**Leo:** And you win bitcoins, yeah, I guess so.

**Steve:** Oh, interesting.

**Leo:** And then this bitZino does poker, blackjack, craps, roulette. So 3.2 million wagers in the second half of last year.

**Steve:** There was actually an episode of "The Good Wife" that I talked about, I think it was probably last year, where it was about Bitcoin.

**Leo:** What? Really, wow.

**Steve:** The topic was bitcoins, and they were arguing in court whether it was a currency or a commodity because the laws differed depending upon whether it was one or the other. And it was really interesting.

**Leo:** No wonder you like the show. Holy cow.

**Steve:** Yeah, it's hit the mainstream.

**Leo:** Yeah.

**Steve:** Oh, I'm now caught up. I've seen all of the first three seasons that I had on disk, and all of this season that my media center has been sucking in all season, even though I hadn't watched one. I am now 100 percent current.

**Leo:** That's what he really did last week, folks. He watched TV. Admit it, admit it. Busted.

**Steve:** I can get my life back now. As long as I don't crack open another Peter Hamilton multiple...

**Leo:** I downloaded the audiobook of that new Peter Hamilton. Have you read it yet?

**Steve:** No, no, no. I do have work I have to get done.

**Leo:** I've got some studying to do, actually. But maybe after that...

**Steve:** Football.

**Leo:** Football study.

**Steve:** You've got to be able to know what's going on if you're going to go to New Orleans.

**Leo:** Yes. "Take your eye off the ball," it says. That's how you pay attention. I have no idea. Excuse me. I have something caught in my throat here. Maybe it's just the Java software.

**Steve:** So, many people - I'll talk while you go fix your throat, Leo.

**Leo:** Okay. Please.

**Steve:** Many people have asked for a sort of a summary of Kaspersky's big revelation of a little over a week ago because this was on my radar for the last podcast, but there just wasn't time to cover it. So I pushed it to this week. And that is the discussion of so-called "Red October," named after the famous movie, "The Hunt for Red October." Kaspersky Labs discovered, at the request of one of their unnamed clients, they discovered another massive, longstanding - and by that I mean since 2007. Some of the files that they've located are date stamped, and they have reason to believe that those dates are valid. So five or six years there has been in place a very substantial spying network. It tends to be focused on embassies and diplomatic facilities, largely in Europe and in the Middle East.

They believe that the code originates with Russian-speaking coders due to various hints throughout the code, some subtle, some not very subtle, like there's a batch file that immediately changes the code page which is the way Windows interprets characters into a code page that allows Cyrillic characters. So that's rather obvious. But it's littered with little tiny subtleties that say the coders were Russian. Now, that's of course circumstantial. You might imagine somebody else could have deliberately, like, chosen Russian-speaking programmers who are not in Russia to do this. But we don't know.

By analyzing everything they've been able to find, infecting machines, watching how they work, creating sinkhole servers, that is, their own targets - I'll explain how they did that in a second - they learned a lot about it. All of the attacks start by either being phishing or watering hole attacks, meaning that either email is sent to somebody they want to gain intelligence from, or websites which are not secure, which are often visited by people whom they wish to gain intelligence from, will be altered with well-known exploits. It's significant that this isn't, from what they've seen, these are not employing the latest cutting-edge, zero-day, what's your Java flaw-of-the-hour sort of problems. But often very old exploits are being used.

For example, once a machine within a victim network becomes infected, it uses what they describe as a "lateral scan," meaning within that layer of the network, meaning an Intranet scan, using the same old attack that the Conficker worm used. And it's MS08-067. Well, the MS08 tells us how old it is. That was from 2008. And so what they are leveraging is the fact that probably in this geographic region, maybe more than others, there are many, many fewer Windows machines being kept current with security. They may not be legitimate. They may not be registered. They may not have access to Windows update mechanisms.

For whatever reason, what they've seen is that this network is reusing relatively old exploits, but they are being highly effective. From the phishing exploits that are used in email, and in some cases bringing up pages on the web, they're exploiting three or four well-known Word and Excel vulnerabilities which have long since been patched in the West, and anyone who has access to Microsoft's Second Tuesday of the Month patching. But they're being very effective.

In every case the first infection runs what's now being called generically in the trojan industry a "dropper" because it drops a file onto the system. The dropped file then contains the payload, which goes out and finds the command-and-control system, hooks into it, obtains instructions, URLs and so forth for downloading other modules. Naturally, in any facility like this, which is going to be long-lived, and in this case from '07, we're talking five years or six years this thing's been around, it inherently has to adapt.

It needs to remain as stealthful as it can so that it's not exposed. But also it can't just be some code written in '07 which is still going to be doing what they want in 2013. So it's inherently modular, and it's sort of organic in that, as they look at different systems, they realize, oh, here is an instance of older code that's still part of the same network that hasn't updated itself compared to the newer code. One of the things they discovered is that, within each instance of this initial sort of hub or root, there will be three domain names hard-coded in. For example, in one instance that Kaspersky shows, there was nt-windows-online.com, nt-windows-update.com, and nt-windows-check.com. So sort of innocuous looking. I mean, they're not named "werehackingyousucker.com." It's like, if you were looking, and you saw, oh, look, that's, I mean, who knows what Windows does anymore. So that's sort of believable.

And in general they found about 60 domains that have been registered to this network over time. When the thing starts up, it verifies that it has Internet connectivity by trying to connect to the three legitimate websites, update.microsoft.com, www.microsoft.com, and support.microsoft.com. If it's able to do that, it decides that it currently has connectivity, and it does a number of very sophisticated things.

For example, they divide the modules into those they call "offline" and "online," the distinction being that the so-called "offline" modules, even though they do communicate on the Internet, they will write things to the hard drive, like the registry and the log files and so forth; whereas the so-called "online" modules deliberately never record anything. So they're, like, running within a different - think of it as a security boundary where they might be more vulnerable to being caught, so they behave themselves against monitoring. They're, I mean, they're just as nasty as anything else, but they never touch the registry. They never touch the hard drive. So again, this demonstrates some real sophistication in the overall architecture.

**Leo:** You know what this reminds me of a little bit is Flame.

**Steve:** Yeah.

**Leo:** Which was the government-sponsored, our government probably sponsored, cyberwarfare tool. Do you think this is the Russians' version of Flame? I mean, it's the same age. It sounds like it's very well written.

**Steve:** Yeah, I mean, this kind of - these level of details demonstrate that this sort of cyber espionage is not science fiction. We've read several of the books that talk about it. But this is actually going on.

**Leo:** This is a Mark Russinovich novel in real life.

**Steve:** Yeah. They have pursued this network to the best of their ability. The problem is that there are chained encrypted proxy servers which have even now prevented Kaspersky Labs' detection of the root master command-and-control server. So they can see what's...

**Leo:** Do you think this is private enterprise? This sounds awfully governmental.

**Steve:** It does sound governmental. This is, I mean, it could be a private subcontractor.

**Leo:** Well, oh, yeah, I mean, and it could be the Russian mafia.

**Steve:** A lot of money's been spent.

**Leo:** Seems like this...

**Steve:** Well, yeah, and they're - looking at their targets - I'm not going to enumerate. If anyone's interested, just put "Red October" and "Kaspersky" into Google, and you'll find their pages because they've dissected this thing down to the molecular level. And there are things like listings of all of the embassies, spread out by geographic region. And the geography tells a story, too. That is, since this is a so-called spear phishing or watering hole attack, there's reason to believe that this isn't like Code Red or Nimda that just promiscuously scanned the Internet and infected everybody they could. The point is you want to keep your head down if you're running a network like this. You want to stay off the radar. The last thing you want is Kaspersky to get wind of it, as they always seem to, because then you're blown.

So this thing for five years or six years has stayed under the radar. It's been operating. And so one of the ways they've done that is to make the infections highly selective. This isn't just going, registering on everyone's scanners all over the Internet. They're selectively infecting specific individuals or offices or organizations from whom they wish to gain intelligence. And they are absolutely wanting not to get detected.

So one of the very cool things about this is that Kaspersky found - remember I

mentioned there were three domain names burned into every sort of kernel of this malware. They looked at, they collected enough samples of it that they found five domains whose registrations had expired: shellupdate.com, msgenuine.net, microsoft-msdn.com, windowsonlineupdate.com, dll-host-update.com - oh, I'm sorry, it was six - and windows-genuine.com. And they registered them. So they registered the domains that had been...

**Leo:** So clever.

**Steve:** Yes, that had been retired and expired. And they are now receiving tens of thousands of connections...

**Leo:** Holy cow.

**Steve:** ...from infected machines to their what they call "sinkhole servers."

**Leo:** Holy cow.

**Steve:** Isn't that cool?

**Leo:** This is, oh, man, this is like a spy novel. This is great stuff.

**Steve:** It is neat. So they set up servers on their own IPs, or probably some that are not associated with them, just for safety. And they registered those domain names which were in the spyware, but which no one had bothered to renew. That allowed them to acquire them. They pointed those domain IPs to their machines, and they started getting calls from older malware that was still in place. And then that allowed them to continue their investigation. So anyway, that's the story about Red October. It is a long-lived, keep its head down, try not to get discovered. It's using old exploits which are still startlingly effective, despite the fact that they're, in the case of an '08 exploit, that's going to be, what, five years old.

**Leo:** This is for sure - I've got to think it's the Russian government. Wow.

**Steve:** It is, well, again, once you discover it, you can see who they've been targeting.

**Leo:** Yeah. Embassies.

**Steve:** And they're targeting...

**Leo:** It's governments.

**Steve:** Yes.

**Leo:** And by the way, you know they shut it down within hours of this report.

**Steve:** Yup.

**Leo:** So. This reminds me of the Anderson tapes, where...

**Steve:** And so that's the other that happens, of course, is that, because Kaspersky knows disclosing it is going to kill it, they wait to tell anybody until they've gathered all the information that they can. And then, when they finally are ready, they go public, and the thing instantaneously dies.

**Leo:** Like "The Bourne Identity." We're shutting this down. Shut it down. Terminate with extreme prejudice.

**Steve:** Now, we talked, boy, it's maybe been a year, about the really egregious mistake that the TRENDnet webcams made. Remember that those were webcams that people were putting on the 'Net that anyone had public access to. The webcam acts like a little web server. And when you connect to a web server, the web server identifies itself. Well, that meant that you could scan port 80, if it was port 80. I didn't go back and refresh my memory. But you could scan whatever port these things were on. They may not have been standard HTTP port 80 servers. But you could easily scan the 'Net for whatever port they were listening on and inquire of them whether they were a TRENDnet webcam. If they were, it was trivial to start receiving from it the imagery that it was sending.

Well, what appeared on the 'Net, it might have just been yesterday, was really frightening. And it's been taken down. Gizmodo wrote about it. The Verge wrote about. I mean, it caught people's attention because what you got was a map of the world from Google, with all those little red sticky pin icons that Google Maps uses, for the geographic location of every one of these still open, a year or however long it's been since I last - since this was in the news and discovered, every single one of them that had been found. And you could click on any of these little sticky pins, and it opened a window, and you were looking into someone's bedroom.

**Leo:** Oh, that's handy. Find your neighbors.

**Steve:** I should have grabbed a screenshot of it yesterday. What's up now, when I went there checking the link for the podcast, I briefly saw the map, and then it disappeared with a statement saying that Google was no longer servicing Google Maps for this domain. So the outrage over this immediately went to Google, and Google shut down their access to the Google Maps API. Oddly, it still comes up briefly and then gets covered up by this message. So it seems like it's probably possible to work around it.

But in any event, if you disable scripting, and I still had mine initially on from temporarily allowing it yesterday in NoScript, if you disable scripting, then you get a static page that just shows what I'm talking about. So if anyone's curious, it's cams.hhba.info. Again,

that's cams.hhba.info. If you go there with scripting enabled, you may briefly see the real map, and then it will disappear. Yup, there it is, Leo, you're showing it in the video. And if you disable scripting, you can see just a static slide that they've put up saying that the site is no longer live, and they have no intention of bringing it back.

**Leo:** Obviously they just wrote a little script.

**Steve:** Yeah. Oh, but, I mean...

**Leo:** Of course this hasn't fixed the problem with the cams. It just means it's harder to find them.

**Steve:** Correct.

**Leo:** It's not even that hard. A Google search finds them; right?

**Steve:** Oh, it's trivial. Yes, I mean, it's really disturbing. And it's funny because TRENDnet immediately after this said, oh, well, we've notified everyone who has our cameras that they need to update their firmware. Well, here's a picture of it, folks.

**Leo:** Yeah. See if you're on the map. Geez, Louise.

**Steve:** Yeah. Oh, big improvement. Okay. Now, the other big news is something that I'm so confused about what I can say that I have to err on the side of not by mistake saying anything I can't say.

**Leo:** Now I'm intrigued.

**Steve:** Yes. Well, this is about Google and YubiKey. And I alluded to a meeting that I had with Stina. She was down in Southern California a few months ago. And that was a reason that I did a podcast on near field technology, because it was due, but I decided, okay, we need to understand what near field technology is. And what sort of leaked out ahead of schedule was that some guys at Google posted, or presenting a paper toward the end of this month, and there's not much month left, so it's going to be soon, that will be in the - it'll be published in the IEEE Computer Society proceedings. The paper is titled "Authentication at Scale."

And so what I can say is that Google is working on solving the authentication problem, and YubiKey is involved as a vendor. So Google is playing with YubiKeys. And I will be able to say more as more is known. I've not yet read the paper, although I spent an hour on the phone yesterday with Stina. And we jumped back and forth so much between this is off the record, this is on the record, that I have no idea what...

**Leo:** That's why I never do anything off the record, because I can't keep track.

**Steve:** Yeah. So...

**Leo:** There is an article, and maybe this will help you because this is, once it's public, then it's public. There's an article in The Verge saying Google wants to ditch passwords, let you log in with the ring on your finger.

**Steve:** Yes. And Wired.com has one, and technology review had one. So there has been Google's alternative to passwords.

**Leo:** As Wired reports, Google envisions a new form of authentication that would let you quickly sign websites with the help of a minuscule USB key. Its researchers have been fiddling with YubiKey graphic cards in particular.

**Steve:** Yes.

**Leo:** And found it only takes a few modifications to Chrome to get a seamless login process running smoothly.

**Steve:** Okay, good. In that case I can say that the cool thing here, the reason I just went crazy when I met Stina at the top of the escalator at the RSA security conference those years ago was that she said - I'm walking around, and no one knew who she was, and she'd been kicked out the booth that she had a commitment to share with some deadbeat company. And so she said, "Are you interested in security?" And here we are at the RSA security conference, so that was a good bet.

**Leo:** A good guess.

**Steve:** Unless I was just there for the buffets or something. So I said yeah. And, I mean, she didn't know who I was. And so she said, "What about authentication?" I said, oh, yeah. And so she says, "This is a one-time password that emulates a keyboard." And, I mean, I instantly knew as much about it as she did because, I mean, that's - really good inventions are like that. And it was like, oh, my god, this is fantastic. And so but the beauty of it was, because it was a keyboard emulator, it was a USB keyboard in a little tiny dongle, it worked without software. It typed for you, into whatever machine you plugged it into, end of story. One-time password. Brilliant. So now, okay, I think I can...

**Leo:** So there is an article. Stina's written on her YubiKey blog at Yubico.com an article describing Google's vision and how YubiKey would be part of that.

**Steve:** And the abstract from Google's IEEE article says - this is their "Authentication at Scale." They said, and this is public: "In working to keep cloud computing users' data

safe, we observe many threats - malware on the client, attacks on SSL, vulnerabilities in web applications, rogue insiders, espionage - but authentication-related issues stand out amongst the biggest. When trying to help hundreds of millions of people from an unbelievable variety of endpoints, attitudes, and skill levels, what can possibly displace plain old passwords? No single thing, nothing overnight, and nothing perfect. A combination of risk-based checks, second-factor options, privacy-enhanced client certificates, and different forms of delegation is starting to find adoption towards making a discernible difference."

Now, Stina's vision, that is well known to anyone who knows Stina, is they don't want to own anything. They don't want anything to be proprietary. They're completely happy to compete in an open market. She wants everything to be standards. She wants everything to be open. She wants no single point of failure. And this collaboration with Google is really exciting because what could potentially happen is, by making some changes to Chrome, they maintain the zero friction usage, zero software installed.

If you use Chrome - and I should mention, not the current YubiKey. They'll make deals for people who have existing ones to upgrade to a next generation because there is some really good stuff in the next - in the YubiKey that works with Google. So we're not going to be able to use the same one. But the reason is that the technology needs to change a little bit. In return for that, though, something really amazing happens.

So what would happen is, all that Google would have to do, after they take this out of pilot, is say, okay, now everybody can use this. And that'll put pressure on any non-Google browser to support this because, for obvious reasons, this thing really could be the thing that catches on. So anyway, it's very exciting. And it's moving faster than people expected.

**Leo:** I love the idea. I mean, I presume it would be the second-factor authentication, not primary.

**Steve:** It's strong enough that it would be up to the user. But what they've got is they've come up with a way of essentially achieving those goals. The problem with VeriSign identity protection, unfortunately, is that it's VeriSign, and it's expensive. And the people who get - not the people authenticating, but the people being authenticated to, pay per authentication a stiff price to VeriSign. And so there's that problem. Then you have the OneID people who say, oh, yeah, we can get rid of passwords completely. It's like, well, yes, if you change the entire infrastructure of the world, you could do it differently. And they want to own it. It's all proprietary. I've asked them for security documents, and they won't ever tell me how it works. So it's like, well, okay, good luck with that.

So the only way we know, the only way this can, any solution can succeed, is if it isn't owned by any one company; if it's a set of standards that are strong enough to really work. And, for example, you could have multiple tokens. In Stina's vision, you buy these things at 7-Eleven or Safeway, in the same way you buy prepaid credit cards. These things are inexpensive. So you use one as your master identity. You could have a couple more for pseudonymous identities. So you can have all of the flexibility that we have now, yet a physical token, a physical hardware token which would both be like the YubiKeys that are now around, both near field and USB, so that it works on your phone, if you've got a USB-enabled phone. And I have to think that Apple's going to have to fix, I mean, I'm sorry, if you have an NFC-enabled phone. I have to think that Apple's going to be fixing that with the next release of the iPhone because Samsung is just having a massive party at their expense.

**Leo:** And it's so awesome. And there's so many things you can do with it.

**Steve:** Yeah. So anyway, that's just - Google has demonstrated obviously their clout, and they've got a browser that people really like. And I'm wishing that it weren't burning so much of my RAM every time I fire it up, but so be it. But...

**Leo:** You use Chrome? So you're using Chrome now as your preferred browser?

**Steve:** No. I fire it up when I'm using the Google Groups because their own groups work just a little bit better in Chrome than others. I just - I'm still in love with Firefox. I'm still a Firefox guy.

So Ed Bott did a blog posting whose link I tweeted, I think yesterday, which is really annoying [zd.net/WTCvni]. Ed discovered that, when you are upgrading Java - and he talked about what I've often talked about. He was talking about the Ask Toolbar.

**Leo:** Oh, this thing, yeah.

**Steve:** Yes. Now, again, it's checked by default. If you are a person who just clicks the Next button on the wizard in the lower right, you will leave it on, and you will get this thing installed.

**Leo:** So frustrating.

**Steve:** But get a load of this. When it comes up to its confirmation of saying we're done, it'll say "Java has been updated, and the Ask Toolbar has been installed." Now, you might go, what? No, oh, no. No.

**Leo:** No.

**Steve:** So a smart person, I mean, any of our listeners would immediately go into the Control Panel and look for the Ask Toolbar in order to say no, thank you, uninstall. Leo? They put a 10-minute delay in.

**Leo:** Oh, that just sucks. That just sucks.

**Steve:** So it sits in the background for 10 minutes.

**Leo:** Oh, they just are evil. What is wrong with Oracle?

**Steve:** And then installs itself.

**Leo:** Oh, that's just - that's malware. Sorry, that's malware.

**Steve:** Is that unbelievable?

**Leo:** That's malware. I'm sorry, that's not okay.

**Steve:** It is. It's unconscionable. So someone realizes that they didn't remember to uncheck it and says, oh, crap, let me get rid of it, and it's not there. And they think, well, okay, something happened.

**Leo:** It didn't install.

**Steve:** It didn't work. So, whew. And then it sneaks in 10 minutes later.

**Leo:** Revolting. Revolting. I think, well, the headlines are "Oracle's trying to kill Java." I mean, come on, Oracle. What the hell? Oh, geez. Well, this does kill Java. That's fine. Bye bye, Java.

[Talking simultaneously]

**Steve:** It's having a bad time lately. Ah, well. So a quick note. Google added a service that someone tweeted to me. I wanted to share it with our listeners because there have been websites around from time to time that show you what your IP is. Well, now you don't have to go to a website. You just type into Google, what's my IP? And the first thing that comes up is your IP. Oh, here's your IP, by the way, and then the standard search results for what's my IP. So that's very cool. Just an FYI, a little tip for our users who care about IPs.

Oh, and you do want to make sure you do it over HTTPS. I don't know for sure, but that's why ShieldsUP!, my own port scanner, uses HTTPS, and that is to bypass ISP caching proxies because that's the IP that nonsecure connections see when you're behind - when you're in an ISP who's using a caching proxy. You don't really want the IP of the caching proxy, which is where the requests go to Google, and so the IP that Google sees. Only if you are over SSL are you able to avoid the non-SSL caching proxy.

Now, because we've got so much to talk about this week, and it's involved, I'm punting my discussion of MegaUpload, aka Mega...

**Leo:** Oh. Because there's some real concern about how secure is it. And they claim AES, but they're sending, I mean, it's an interesting question. Okay.

**Steve:** It is. And so it's involved enough that I didn't want to do it. I didn't want to shortchange anyone. So next...

**Leo:** Give us the bottom line. Should we avoid MegaUpload?

**Steve:** I don't know for a week. I won't know till next week. I mean, as you said, it is so involved, and I just - I had it on my list of things to get to. I couldn't get to it. But I will definitely have a complete readout on it. I do know that they're defending themselves against all this flurry. There's been a whole bunch. And so I need to read what everyone is saying, read their defense, and then I'll process it and let everyone know what I think.

**Leo:** Great, thank you.

**Steve:** A week from now.

**Leo:** A week from now.

**Steve:** Okay. So I also tweeted this. It's always interesting when I tweet things because people assume that I'm promoting what I'm tweeting, when in fact I'm making sure that people who care know. So for people who care, and our listeners know how I feel about Windows 8, the Windows 8 Pro special $40 upgrade offer ends on February 1st, and then it goes up to $199 [bit.ly/13NDlaZ]. So if it...

**Leo:** I bet it doesn't. My prediction. Just going to make a prediction here.

**Steve:** Well, okay. It's a Windows blog. There's the link there.

**Leo:** No, they say it ends. I'm just saying, let's see what happens then.

**Steve:** Yeah, it's like...

**Leo:** Because nobody's buying this piece of crap.

**Steve:** It's like Windows 7 security. Well, in fact XP got extended because too many people are still using XP.

**Leo:** Nobody's buying Windows 8. And making it 200 bucks is not going to help it sell any better.

**Steve:** Oh, goodness.

**Leo:** So I have a feeling that we will see a - maybe they'll extend that offer just a

little longer.

**Steve:** So of course you can imagine people said, wait a minute, you care about that? I thought you were a developer, blah blah. Yes, I get all of this stuff as my $2,500 a year I pay Microsoft for the privilege of having access to all these things.

**Leo:** You're performing a service for our audience.

**Steve:** Yes. I just wanted people to know, if they were thinking about it, in case it does go up on February 1st to $200, you can still get it, that beautiful shiny...

**Leo:** You know, the more I use it - I've been reviewing Windows 8 laptops. The more I use it, the more I wonder what the hell they were thinking.

**Steve:** I know.

**Leo:** It is awful.

**Steve:** Leo, how about solving security? How about...

**Leo:** Well, they can do both. They can do both. It's not an either/or.

**Steve:** No, I mean, they've got resources. They're - anyway. Done. Yeah. They're not spending their resources where I would have them spend them. Now, I have to share this. I'm a little worried that this got enough coverage that it won't be a surprise for every listener because Friday I was mentioning it to Jenny, and her mom had run across the story. So it's like, okay, well, if Jenny's mom knows about it, maybe it's no longer a secret. But I'm sure many people haven't heard about it, and it's just too wonderful not to share. So this was posted on January 14th on the Verizon RISK Team security blog. And I'm just going to read it because there's nothing I could - paraphrasing it won't work, and it's just wonderful. So this is a case study, the moral of which was the title of the blog posting: "Proactive Log Review Might Be a Good Idea."

So they said: "With the New Year having arrived, it's difficult not to reflect back on last year's caseload. While the large-scale breaches make the headlines and are widely discussed among security professionals, often the small and unknown cases are the ones that are remembered as being the most interesting from the investigators' point of view. Every now and again a case comes along that, albeit small, still" - are you chuckling in the background, Leo?

**Leo:** Yes.

**Steve:** "...albeit small, still involves some unique attack vector, some clever and creative

way that an attacker" - huh?

**Leo:** No, I'm just reiterating. Clever, creative, yes.

**Steve:** "...clever and creative way that an attacker victimized an organization."

**Leo:** Shocking.

**Steve:** "It's the unique one-offs, the ones that are different, that often become the most memorable and most talked about amongst the investigators. Such a case came about in 2012. The scenario was as follows."

**Leo:** Such a good story [laughing].

**Steve:** You couldn't make this up, Leo. This is so good. "We received a request from a U.S.-based company asking for our help in understanding some anomalous activity that they were witnessing in their VPN logs. This organization had been slowly moving toward a more telecommuting-oriented workforce, and they had therefore started to allow their developers to work from home on certain days. In order to accomplish this, they'd set up a fairly standard VPN concentrator approximately two years prior to our receiving their call. In early May 2012, after reading the 2012 DBIR" - whatever that is [Data Breach Investigations Report] - "their IT security department decided that they should start actively monitoring logs being generated at the VPN concentrator.... So they began scrutinizing daily VPN connections into their environment. What they found startled and surprised them: an open and active VPN connection from Shenyang, China. As in, this connection was live when they discovered it.

"Besides the obvious, this discovery greatly unnerved security professionals for three reasons: [One] They're a U.S. critical infrastructure company, and it was an unauthorized VPN connection coming from CHINA" - in all caps. "The implications were severe and could not be overstated. [Two] The company implemented two-factor authentication for these VPN connections, the second factor being a rotating token RSA key fob. If this security mechanism had been negotiated by an attacker, again, the implications were alarming. [Three] The developer whose credentials were being used was sitting at his desk in the office.

"Plainly stated, the VPN logs showed him logged in from China, yet the employee is right there, sitting at his desk, staring into his monitor. Shortly after making this discovery, they contacted our group for assistance. Based on what information they had obtained, the company initially suspected some kind of unknown malware that was able to route traffic from a trusted internal connection to China, and then back again. This was the only way they could intellectually resolve the authentication issue. What other explanation could there be?

"Our investigators spent the initial hours with the victim working to facilitate a thorough understanding of their network topology, segmentation, authentication, log collection and correlation and so on. One red flag that was immediately apparent to investigators was that this odd VPN connection from Shenyang was not new by any means. Unfortunately, available logs only went back six months, but they showed almost daily connections from

Shenyang, and occasionally these connections spanned the entire workday. In other words, not only were the intruders in the company's environment on a frequent basis, but such had been the case for some time.

"Central to the investigation was the employee himself, the person whose credentials had been used to initiate and maintain a VPN connection from China. Employee profile: Mid-40s software developer versed in C, C++, Perl, Java, Ruby, PHP, Python, et cetera. Relatively long tenure with the company, family man, inoffensive and quiet. Someone you wouldn't look at twice in an elevator. For the sake of case study, let's call him 'Bob.'

"The company's IT personnel were sure that the issue had to do with some kind of zero-day malware that was able to initiate VPN connections from Bob's desktop workstation via external proxy and then route that VPN traffic to China, only to be routed back to their concentrator. Yes, it is a bit of a convoluted theory and, like most convoluted theories, an incorrect one.

"As just a very basic investigative measure, once investigators acquired a forensic image of Bob's desktop workstation, we worked to carve as many recoverable files out of unallocated disk space as possible. This would help to identify whether there had been malicious software on the system that may have been deleted. It would also serve to illustrate Bob's work habits and potentially reveal anything he inadvertently downloaded into his system. What we found surprised us: hundreds of deleted PDF invoices from a third-party contractor/developer in - you guessed it - Shenyang, China.

"As it turns out, Bob had simply outsourced his own job to a Chinese consulting firm. Bob spent less that one fifth of his six-figure salary for a Chinese firm to do his own job for him. Authentication was no problem: He physically FedExed his RSA token to China so that the third-party contractor could log in under his credentials during the workday. It would appear that he was working an average 9-to-5 workday. Investigators checked his web browsing history, and that told the story.

"A typical workday for Bob looked like this: 9:00 a.m., arrive and surf Reddit for a couple of hours. Watch cat videos. 11:30 a.m., lunchtime. 1:00 p.m., eBay time. 2:00-ish, Facebook updates and LinkedIn. 4:30 p.m., end-of-day update email to management. 5:00 p.m., go home.

"Evidence even suggested he had the same scam going across multiple companies in the area. All told, it looked like he earned several hundred thousand dollars a year, and only had to pay the Chinese consulting firm about 50 grand annually. The best part? Investigators had the opportunity to read through his performance reviews while working alongside human resources. For the last several years in a row Bob received excellent remarks."

**Leo:** So what is the problem? He's doing great.

**Steve:** Yeah. "His code was clean."

**Leo:** Very good stuff.

**Steve:** It was "well written and submitted in a timely fashion. Quarter after quarter, his performance review noted him as the best developer in the building."

**Leo:** Wow. Oh, my. It would only have been, I mean, if he hadn't given them access to the VPN, I don't - is it illegal to outsource your contracting work?

**Steve:** I bet you that the employment agreements probably had something to say. If not, it'll be immediately amended to add that.

**Leo:** He did what he said he would do. He provided the work. It was well done. It was high quality.

**Steve:** Ow.

**Leo:** Probably FedExing the VeriSign token wasn't so cool. That's, I mean, basically "spyferret" nailed it. Bob just promoted himself to management.

**Steve:** [Sputtering]

**Leo:** Sorry.

**Steve:** I just took a drink of coffee, Leo.

**Leo:** That's pretty good.

**Steve:** I almost spit it out on the monitor.

**Leo:** Now he's a manager. That's what managers do. I would have done it if I could have gotten away with it.

**Steve:** Wow.

**Leo:** If you see a Chinese guy hosting the show next week, you'll know what happened.

**Steve:** Oh, no, Leo [laughter]. So, okay. So this little dip into geek humor, I have a couple things, was prompted by an acronym for Java that we will wrap with. First, though, I did a little poking around because, as we were saying at the top of the show, I've always - my favorite thing for Fiat was always Feeble Italian Attempt at Transportation. So it turns out there's a website of these. And I just put some - oh, I put "car acronyms jokes" into Google, and it found a website for me where I found some I'd never seen before that I particularly - and I don't know why these really - I just - I do like puns, and for some reason puns, I have heard, are considered the lowest form of humor. But they just always tickle me. So Buick as Big Ugly Indestructible Car Killer I

think is pretty good. But I have to say I found some for BMW that just give me a kick. Okay, so BMW stands for Bought My Wife. Brutal Money Waster. Born Moderately Wealthy. Brings More Women. And Broke My Wallet. So anyway...

**Leo:** People are quite inventive. There's a lot of...

**Steve:** Oh, they go on. Now, some are really awkward, like there was Yamaha or...

**Leo:** The longer ones are harder.

**Steve:** ...Toyota, yeah.

**Leo:** Honda: Had One, Never Did Again. Hang On, Not Done Accelerating. That's a good one [laughter]. That's from LotsOfJokes.com.

**Steve:** So Java's official name henceforth.

**Leo:** Yes.

**Steve:** Because this is just too good.

**Leo:** Yes.

**Steve:** JAVA stands for - and thank you, whoever tweeted it - Just Another Vulnerability Announcement.

**Leo:** Oh, boy. Wasn't there another one, a new one? For some reason I think there was a new zero-day, unless, you know, you get - after a while it's just a blur.

**Steve:** Oh, we're just dizzy. And I did like a tweet that I saw. I just love - this demonstrates the caliber of geek people that we have. Actually the Twitter handle is @LethalDosage1, the numeral one. And @LethalDosage1, whose name is not in his profile, otherwise I would use it, said: "@SGgrc Steve, you almost have 32768 followers. I hope you're using unsigned!"

**Leo:** Otherwise you go to -1.

**Steve:** Wonderful.

**Leo:** That's good. Somebody's paying attention. Wow.

**Steve:** Yes, yes, yes.

**Leo:** That's funny.

**Steve:** Yup. Yeah. For those who are not programmers, if you store a number in a 16-bit representation, you can have it be unsigned, in which case it's a simple quantity which can have any value from zero to 65535, which is the maximum, which is 2^15 plus 2^14 plus 2^13 plus 2^12, all the way down to the zero, which is 65535. But you may want to be able to represent negative values, in which case the highest order bit, the bit 15 if you number from zero, that is the so-called "signed bit." So if it's on, that is, if it's a one, then the rest of the bits represent a negative value. So what would happen at 32767 followers would be the maximum positive number you could represent with a signed 16-bit quantity. You add one more, and that flips it over, turns the signed bit on, and actually then it becomes -32768, and it starts counting backwards towards zero from there. So anyway, a nice observation. I thought that was - I got a kick out of that. That's good.

Okay. Also in my Twitter stream today, someone tweeted me a neat picture. His name is Troy Thompson, and he sent me a flip.it link [flip.it/bdM2b] which is a picture of a lab - and Leo's showing it on the video right now - of a huge table of laptops all running SpinRite at once. And the caption says something about, I think, they're running SpinRite on laptops which will be raffled off, and so they will have the best, most tested hard drives ever. Anyway, so I saw that. I just tweeted back to Troy and said, "Hey, thanks for that, that's a cool photo." And he sent me a link to a review that he wrote of SpinRite in '07. So it's dated. But of course, as we have seen, hard drive technology isn't changing very much. So I just thought I'd - I'm going to excerpt a little bit from this.

The title is "SpinRite - Review by a Fan." And so he said, "The SpinRite hard drive maintenance and recovery utility software by Steve Gibson of GRC.com has been around for years. I first encountered SpinRite during the summer of my sophomore year of college. This is my personal unbiased review of my SpinRite experiences." And I've snipped out a bunch there, but he's talking about drives, and he says, "Hard drives are far from perfect. Instead, they are excellent at hiding the fact that errors happen, and happen frequently. While the operating system is told the drive is flawless, the drive itself is frantically performing error correction on the fly" - which we all know is true, we've been talking about that recently - "to compensate for the fact that we're rapidly approaching the point where laws of physics are limiting what can be done with magnetic media." Which of course we talked about. Inherent economic pressures force companies to always store as much as they possibly can and probably go a little farther than they should.

He says, "By the time Windows System Event log throws errors from ATAPI, disk, or other devices about 'error during paging operation' or such, it's nearly too late. The drive has run out of ways to compensate for problems, and has finally told the upper-level operating system, 'I'm dying here.' Kiss the drive goodbye. Or not. Enter SpinRite, stage center.

"SpinRite, presently at version 6.0" - as it still is today - "as I write this is still around. With over 16 years of history under its belt" - and now we're at 20-plus - "SpinRite has

achieved a lifespan nearing that of Norton Utilities and other classic disk tools." Actually I think we're now the last man standing in that regard. He said, "And I must confess" - and he has in bold italics - "it works precisely as described. There are those who accuse SpinRite's author, Steve Gibson, of hyperbole and scare tactics, and summarily state that SpinRite is mere snake oil. Say what you will of Steve's mannerisms, he is by far one of the most talkative tech podcasters I've heard."

**Leo:** Well, he just hasn't heard me.

**Steve:** Oh, no, I'm sure he's heard us both. I'd never have been doing tech podcasts without you, Leo. He said, "SpinRite's success is not attributable to slick marketing or fakery. It recovers data when nothing else can. "One more thing for the geeks: SpinRite is a svelte DOS and Win32 executable weighing in at a mere 170KB, written in pure assembler. In my book, this earns Steve Gibson very high geek points."

And then finally, under "Success Stories," he says, "I have used my copy of SpinRite to fix" - in bold italics - "at least two failed hard drives per year amongst my family. When a friend solicits assistance with a drive recovery, if SpinRite does the job, I urge them to purchase their own copy, too." And of course everyone knows I have no problem with that policy. "At EnvisionWare, we have a corporate site license, bootable from the LAN - very handy! - and have used SpinRite to repair" - again bold italics - "four failed drives per year on average, recovering data which was not recoverable by any other methods available at the SpinRite price point. We have recovered numerous laptop and desktop drives, and we've started running SpinRite on every new system before deployment, ensuring that the drive has been completely worked out before it hits the road.

"I've seen SpinRite run on an unbootable, unreadable drive that was actually making click-of-death type noises, where the internal motor driving the heads across the platters was making a constant clack-clack-clack as it tried to read. After several days of work, SpinRite had recovered enough data to make the file system readable so we could recover the data. The drive was completely dead by the time we finished the recovery."

So, and then he goes on. And he really understands how a hard drive works, talking about sector relocation events and relocated - actually our listeners would care about this. So I'll continue, just finishing up quickly, "What doesn't kill you makes you stronger unless you're a hard drive. A word of warning here. If a hard drive is nearly dead, SpinRite will often warn against running anything except a basic data recovery. This warning should be heeded. If a drive is marginally readable, or is physically damaged to the point that heat or wear would kill it, SpinRite may very well be the last thing the drive sees. This is because SpinRite is best used preventively. In my experience, running SpinRite every few months is a good way to detect far in advance" - he has in italics - "that a drive is going to die."

And of course I absolutely believe that, that when you're running SpinRite, the SMART system is really showing you in an analog fashion how hard the drive is working to recover from problems that it hides. And the harder it's working, it doesn't say it's going to die soon, but it shouldn't start working that hard.

So anyway, continuing, he says, "The most telling predictors of pending hard drive problems in my SpinRite experiences are: SMART Sector Reallocation Events or Reallocated Sectors. In over 90 percent of hard drives requiring SpinRite recovery, there are at least one sector reallocation event recorded by SMART. You can monitor SMART statistics on your hard drives in real-time from Windows using the smartmontools

software provided at smartmontools.sourceforge.net. Note that, due to the way SpinRite works, a marginal drive may trigger sector reallocations during a SpinRite run. This is by design and is part of SpinRite's data recovery method. It's still a warning that the drive is showing early signs of trouble. I'd recommend running SpinRite multiple times at Level 4 until the sector reallocation events stop increasing. And keep your backups up to date."

So he says, "Rising ECC or Seek Error Rates: A notable rise in ECC or Seek Error rates between SpinRite tests over a month-to-month time period can indicate trouble." And that's one thing that SpinRite shows on the SMART monitor page while you're running it. And if those do go up, that's an indication that the drive is having trouble. He says, "My current otherwise-flawless SATA Seagate drive is sailing along in SpinRite now at an error rate of 1,664 seek errors per" - now, he wrote "minute," but it's actually per megabyte are the units that SpinRite shows. That way it's a constant measure. So it's at 1,664 seek errors per megabyte. If that number goes up significantly, like by 25 percent or more, watch out."

And he wraps up with, "Why Don't You Own It Yet? That's the real question. If you're not a SpinRite fan already, why don't you buy a copy of SpinRite?" He says, "There are a few people who should not bother: You have daily backups of all your data, and a spare drive on hand." So he's a little tongue-in-cheek here. "You have RAID controllers or your own SAN, which is backed up." Or, three, "You don't mind losing your data. Seriously," he says, it's that good." So, wow, Troy. Thank you very much.

**Leo:** Nice compliment. That's great. Very nice. And that's the same guy who had the pictures of the laptops.

**Steve:** Yeah. And, well, that's how I found this review. I had never seen the review before. But when I thanked him for posting the cool link of pictures, then he sent me a link to his review. So...

**Leo:** Very nice. Memory hard problems. Not hard memory, memory hard.

**Steve:** Right. So as I said at the top of the show when we were talking about our topic for the end of the show, which is where we are now, the problem is that we are currently using existing crypto technology in a way it was never really meant to be used. Someone sent me, it must have been a tweet because it's short, and I don't think I saw it in the mailbag. So it may not have been a listener. But the person tweeted, "If my bank locks me out after three misses, why do I need a hard-to-use password?"

**Leo:** That's a fair question.

**Steve:** It is, absolutely. And our listeners know that the reason is not that your bank is actually going to sit patiently by while you guess passwords day and night through the web browser interface, which itself imposes a huge penalty in terms of per guess time. The problem is that we see ridiculously, I mean, almost as often as Java is getting a new zero-day exploit, somebody is losing control of their password database. It's happening all the time. So the problem is, if your bank lost access, that is, got hacked, and hackers acquired the hashes - first of all, if the bank weren't hashing the password, if it was just storing the plaintext passwords, well, they instantly have every account is compromised.

But that requires really bad security practices. Most times the bank is hashing the password. So that means that they're storing something which is a one-way function. You can, given the password, you can get the hash. But you cannot go the other way. There's no key. It's not like a cipher where you can decrypt what has been encrypted. The hash is a lossy - it's an information lossy process. It doesn't attempt to retain all the information. It attempts to create a fingerprint of the password. It's like, this password gives you this hash.

And so the point is the bank, if the bank only stores the hash, then they don't know the password. They can't mail it to you. They can't give it to you. The only thing they can do is if you provide it again, they can compare it to the one they have and see if it matches. And if it does, it's almost certain that you are the person logging in. Not absolutely galactically positively certain because it's possible to have a so-called "hash collision," where different things input result in the same output. But the chances of that are diminishingly, vanishingly small. And this is why the password, the hashes have so many bits. If you had a 256-bit output from the hash, you would have to have all 256 bits come out exactly identical. And just statistically, that's ridiculously unlikely. So it's safe.

The problem is that the hashes were designed to be executed quickly. So in answer to this person's tweet to me, the danger is that the bank would lose its hashed password database. And then bad guys could, using big monster arrays of graphics processing units which have been custom programmed to run these hashing algorithms fast, could pour millions, truly millions of passwords in, looking for matches against the hashes. And so that's the problem. It's not that someone is going to guess your password trying to log in. It's that the bank will lose control of its database, and then the bad guys, if you have easy passwords which are going to be found through that massive bulk guessing game using graphics processing units, then someone can log in as you the first time they try and take all your money.

So to thwart this problem, the current best practice is to use something called PBKDF2, which is an awkward acronym for Password-Based Key Derivation Function. The hash we were just talking about is a very weak version of that. A stronger version is to do it many times. The idea being you take a salt of some kind, which is hopefully just a random junk. It's like, for every user account, a random number is - it's called a "nonce" because you just ask for it as just a piece of - it's like a big cookie. It's just a blob of randomness. You append that to the user's password, and you hash it. And you append that blob of randomness so that the same password doesn't always hash to the same result.

So you can put in a user's database, in the clear, in public, this is the nonce that was used. It doesn't really matter whether that's secret or not, though it's convenient if it's secret, but it doesn't have to be. Just the idea is we want to make it something that helps make the outcome unique per user. Then you take the output from that hash, and you may append the nonce to it again if you want, but then you hash it again. And then you take that output and append the nonce to it if you want, and hash it again. And so you iteratively hash this maybe 5,000 times.

Now, then you take the result of that as the user's hash. So what does that do? Well, when the user is logging in and puts their password in, then on the server end it does this same process 5,000 times that it did when the user created their account. So it's going to get the same result using the same nonce, the same number of iterations, same password, out comes the same answer. So you can test it. The reason this is better than only doing it once is that bad guys have to also do it 5,000 times. They have to duplicate what the bank does in order to know what the password is that you put in to get the right thing out. So that slows down the brute-force cracking by a factor of the number of iterations - 5,000, for example. So that's a good thing.

But here's the problem. Hardware is getting faster. It's getting smaller. It's getting more powerful. We've got people boasting about, and we've seen pictures, we've talked about, like, racks of PlayStation PS2s, all programmed to crack things. Hobbyists are building machines that are nothing but graphics processing units with Freon being poured over them in order to keep them cool because they're running so hard and so fast, and they're doing hashing. It may be a crack station, or they may be bitcoin mining, where a hash is the way you mine bitcoins.

The problem is, I mean, this really is a problem. Actually it's a problem for Bitcoin that their so-called "proof of work" in the bitcoin currency system, it's called a "proof of work" because you require a certain amount of work in order to create a bitcoin from thin air. Unfortunately, they used a problem that was not memory hard, that is, a hash. So the problem is that you can scale with more hardware. You buy more graphics processing units, you're able to produce more hashes.

Okay. So we need to talk about parallel processing and pipelining because the weakness of iteration, anything that is iterative can be pipelined, which is a neat trick for speeding things up. Pipelining was first - it first appeared in early minicomputers. The way computers used to work is they had very simple hardware that went through steps. It would fetch an instruction, the next instruction from memory. Then it would decode that instruction to figure out what it should do. Then it would execute that instruction. And then maybe it would store a result back in memory, depending upon whether it was an instruction that was writing to memory or not. It might just add something in the accumulator and not to memory.

But the point is that execution took these three or four - fetch, decode, execute, store, fetch, decode, execute, store, fetch, decode, execute, store - it took those individual steps. And everybody was happy. It's like, wow, we've got minicomputers. Then they of course wanted more speed. And they said, well, this is fine, but we'd like it to be faster, please. And the people making them said, oh, if we make it faster, we can charge more. So we need a next-generation idea.

So they thought about it, and they were already - already, of course, each of these steps was as fast as they could make it. I mean, they were fetching as quickly as they could. They were decoding as quickly as they could. They were executing as quickly as they could. So they couldn't make those any faster. But they realized, hey, those three steps - the fetching, decoding, and executing - they're separate. That is, say we fetch the first instruction, and while we're decoding that, we fetch the second instruction. And then while we're executing the first instruction, decoding the second instruction, we fetch the third instruction. And while we're executing the second instruction, we're decoding the third instruction and fetching the fourth instruction.

So you see what happened, they overlapped those individual stages so that each of those aspects - the fetching part, the decoding part, the executing part - was busy all the time. It used to be that they were only one third busy. The decoder was idle during fetching and executing, and the executor was idle during fetching and decoding. But by overlapping them, they got three times the performance from the same hardware at the cost of a little more complexity. But suddenly this thing was three times faster.

So this was the birth of pipelining. It's called a pipeline because, if you think about it, if you think of, like, fetch and decode and execute as connected boxes, or a pipe, instructions come in, and they get fetched; then they move to the next stage, and they get decoded; then they move to the next stage and they get executed. So they're moving through this pipeline. And when the pipeline is "full," as it's called, then you're finishing an instruction every single cycle, just as you're fetching an instruction every

single cycle, instead of every third or every fourth. So very cool.

But look at the password-based key derivation. You can do the same thing with it. If you use some salt, and you hash it with a password, you get a result. Then what do you do? You do the same thing again. So imagine, instead of just looping that one algorithm, if you're in hardware, you pass that to the next hash chip. And then it passes it to the next chip. And it passes it to the next chip. And you've got, say, 5,000 of these. The way chips are these days, you might have a hundred on a wafer. And so you've got a hundred hash functions on one wafer, and you only need 50 of those in order to get 5,000 hash functions.

Now, the first time you put something in, it's going to take 5,000 moves to get all the way through. But if you put another guess into the front end every single chance, before long, after 5,000 steps, this machine you've made is now spitting out answers at the same speed as a single iteration hash. So we have completely defeated our password-based key derivation function, which is iterative, by - it's called "loop unrolling." We've unrolled the loop into a linear array of hashes, and we've created a pipeline. We put guesses in the front, and nothing comes out for 5,000 cycles. Then results come out every single cycle. And this thing runs at full speed, just like there was no iteration at all. So that's the problem is we're facing a world where people really can, in their living room, in their garage, have hardware that could do a 5,000-iteration password-based key derivation in hardware. And we're back to square one.

Now, they're prevented from doing 5,000 different guesses at the same time. So we've slowed them down a bit. But the point is this doesn't scale well. Or I should say, from the hacker's standpoint, it does scale the way they want it to. It's not sufficiently difficult. So what do we do? How can we prevent that? Well, what we want to do is we want to, standing back a bit, we want to raise the cost of computing a password. So cost comes in different forms. You could have the temporal cost, that is, the cost in time, how much time it takes to do it. And that's what the current iterative PBKDF2 approach solves for us is it makes it more time costly.

But the other aspect is hardware. And that's what we're trying to make expensive. The problem now is that the algorithms we're using are not hardware expensive. They were designed to be fast in software and almost elegant in hardware. And so you can develop, like, field programmable gate arrays, FPGA chips, that are just, bang, create and compute a hash.

So the overall cost is the time it takes times the cost of the hardware. And we know that what is expensive is real estate. On a chip, when you make chips from wafers, the bigger they are, the fewer of them you can put on a large wafer because the wafers are typically big, round, they maybe six-inch wafers that then have the same pattern in a grid, and you throw away the ones that are not whole around the circular edge, and you get all the ones that are square in the middle. Clearly, the smaller they are, the more you get per wafer, and that directly affects the cost. So we want something which is somehow expensive in physical size, something that forces us to use area.

And one thing which uses space which algorithms don't use is memory. Storage, we've done everything we can to make it smaller. The limits of physics have been pushed. Memory takes space. If you need memory, then that's going to make your chips bigger, or you're going to need many chips, and then you're going to need to talk to the memory. And that's going to mean interconnection between your processing and the memory. And suddenly the whole problem of computing something skyrockets, on orders like 2,000 to 20,000 times. And it's future proof.

The problem with something which just uses tiny chips and sees how fast they can go is that they keep getting smaller, and they keep getting faster, and they keep getting cheaper. And so it's cost effective to use more of them. And if you've got any kind of a process that can be done in parallel, then you're in trouble. So what we want is memory hard problems, not hashes. Hashing could be involved, but we want something that we cannot fool. And that's where the crypto comes in. Something that cannot be fooled, where whatever it is we do requires, it requires a huge array of memory, and there's no way to cheat.

And the cool thing is it turns out it's simple. And here's an algorithm. This is sort of a simplified version, but it's enough that you can get your hands around and I can describe over the podcast, and you'll get it, and it's all you need. It works. And that is, you take a - use a hash function. Use whatever we want - SHA-1, SHA-256, whatever. Doesn't matter. And we fill a large region of memory with pseudorandom data derived from the password. So we take the password and hash it. Out comes 256 bits. Store them in the first area of memory. Hash that again. Salt it if you want to, mix the password in again, doesn't really matter. Hash it again, store that. Hash it again, store that. Hash it again, store that. Fill the entire realm of memory. And we're in the gigabyte realm. So we've got gigabytes. Let's do four gigs. Fill four gigs with this pseudorandom data based on the password. So every time this is done with the same password, we'll get the same array of noise.

Now, what we need - so that's part one. What we need is some way to prove that all of that memory was filled and present all the time. That is, we need a system that we cannot cheat, that there's no way to do this in a little scratchpad somehow. So what we do is we then take the password and hash it, the same way we started before. This time we take some piece of that hash, maybe the high end, the middle, the low end, and we use that to address into the array. So we use that as a pointer into the array and take the data there and hash that with the first hash value. The output of that is another pointer into the array. So we look up what's there and maybe XOR it with the password, whatever we want to do, hash that, and take part of that as a pointer again.

And so you can see where we are. What this does is we have filled an array with pseudorandom bits, huge array. Then we are jumping in a pseudorandom pattern throughout that array. And where we go is based on what is stored there. So the only way, when we're done, to have the final result of this is if everywhere we went we found what we expected to find in that location. We don't know in advance any of this except that we know that, when we're finally done, we end up with a value. And we know that any time we put the same thing, the same starting value in, and do this to it, we're going to get the same value out. And we also know that the only way to do that is if that memory is physically statically present in its entirety as we madly jump around it many, many times, getting the data where we land and using that to tell us where to go next, which is all pseudorandom.

So our path through this array is fixed for a given input, totally unpredictable, and we have to have all that memory there. Now you could say, well, you really don't. You could do this massively computationally, where you could instead look up what's supposed to be somewhere and then go back to the beginning and run all the hashes in order to compute what's there and then use that to jump to somewhere else. Yes. You could do this with no memory at massive cost of execution time. And so the idea is that there is a time and memory tradeoff. But by pulling this out of a purely algorithmic mode into a mode that requires lots of memory, the only, I mean, this is already going to be slow. And so it's ridiculously slow if you try to emulate four gigabytes of stored pseudorandom data, not by storing it and then bouncing around through it, but by actually having to iteratively compute the value that would be stored somewhere in that four gigabytes and

then to have to do it again.

So this thing, it solves the problem. There's no way, because you were needing memory, which is physically large, FPGAs can't be employed. It's going to be slow to do it once, but feasible to do it once. And in fact, this is the technology, a variation on this was developed by Colin Percival, who is the author of Tarsnap, that we've talked about for his cloud computing solution. He uses this technology, it's called Scrypt. And it's been adopted by the IETF and will be becoming an Internet - it's on the way, it's on track to becoming an Internet standard in the future as the way to securely protect passwords from brute-force attack. So it's very cool.

Oh, and there is a competing currency to Bitcoin called Litecoin. And Litecoin uses this, too. So I got a little bit of a kick out of it. On the Litecoin pages they talk about how they're using Scrypt for their proof of work. And the advantage to existing bitcoin miners is that your GPUs can be still left minting, trying to mint bitcoin because they're of no use minting litecoins. Your CPU is the only thing that can be used to mint litecoins. And the cleverness, the beauty of litecoins is they've got a non-hardware-scalable proof of work. So somebody with a fast CPU is just as capable of minting litecoins as anybody else because Litecoin didn't do the simple hashing. It's the simple hashing which created this ridiculous bitcoin mining frenzy, where it just isn't worth regular mortals trying to mine bitcoins anymore because we haven't converted our whole living room over to air-conditioned bitcoin-mining systems. So it's very cool.

**Leo:** Wow, that's really neat. And what is this other coin again?

**Steve:** Litecoin, L-I-T-E.

**Leo:** L-I-T-E.

**Steve:** So, yes, there is another digital currency.

**Leo:** But does it make a difference if there's more than one?

**Steve:** I don't know. I mean, I don't know.

**Leo:** It's the same idea as Bitcoin; right?

**Steve:** It's the same idea. Their page talks about some benefits. And they know about Bitcoin. They refer to Bitcoin. They just think their coin is better.

**Leo:** Oh, yeah. They're trying to create an alternative to Bitcoin.

**Steve:** A coin of a different realm.

**Leo:** They say, "A coin that is silver to Bitcoin's gold." Hmm. Wow. Okay.

**Steve:** Yeah. And it's very cool, though, that they used the Scrypt technology, this approach, which a GPU is no use for. You cannot use - you cannot mint Litecoins with a GPU. It just can't solve this problem. Your CPU can. But so the point is, their proof of work, the Litecoin proof of work, based on Scrypt, based on this, this memory hard problem, is a much better thing to ask people to do because it's - see, what's happened is the hash has just been exploited. It's gone crazy. I mean, there are companies that make Bitcoin mining boxes for people who want to do that. And they're always being obsoleted. You buy one today, and it's obsolete in a month because someone came out with a faster way to hash. You cannot - that won't happen.

**Leo:** So you can't cheat on this one. This is no way.

**Steve:** No, exactly.

**Leo:** Well, I'm going to start using Litecoin, then, instead of Bitcoin.

**Steve:** And similarly, you cannot cheat on using this as your password hash function. It's going to take time. And no roomfuls of hardware can make it faster. It's just going to - it's going to be expensive in memory. It's a memory hard problem.

**Leo:** Which is not to say difficult. Is that what the "hard" comes from? Or is it some other...

**Steve:** No, you're right, it's not difficult. In fact, I was just able to explain how it works. So, I mean, I could explain how this works where I'd have a difficult time, well, I guess I have explained how hashes work. But there are things that are way more difficult, that are not hard.

**Leo:** [Laughing] Which is completely different from a hard end, but we won't get into that.

**Steve:** Or as my dad used to tease me with, "Odd, but not peculiar."

**Leo:** You want to know odd but not peculiar? I'm sitting here, I'm watching my mouse move around, people pulling down menus and stuff on the screen, and I can't figure out how the heck, and I rush over to make sure - because it's the new Mac, and I installed it. And I rush over - that's what I was doing when I was a little distracted. I rush over to see if the firewall's on. Yes, it is. Is sharing turned on? No, it's not. What the heck? And then Chad said, "Yeah, I just borrowed your track pad." I was trying to use it. It's a Bluetooth track pad.

**Steve:** Wireless, yes, yes.

**Leo:** Yeah. The mystery explained. Steve Gibson is at GRC.com. Great place to go if you want to find out more about everything he talks about on the show. In fact, if you have a question, go there, GRC.com/feedback, and you can get some clarification about this or any other topic that's on your mind. I know we'll have some Java questions for next week, and probably a few about memory hard. He also does a lot of other stuff there, including, as you know, his SpinRite program. That's his bread and butter. You can get that from GRC.com. And lots of freebies, too. Lots of great free stuff. New one coming soon.

**Steve:** And a new service to be announced on the podcast next week. Our listeners will learn about it first.

**Leo:** Follow Steve on Twitter, @SGgrc.

**Steve:** Yes, make my number go negative.

**Leo:** [Laughing] Wouldn't that be funny if it did? I'm follower No. -1. That'd be very funny. And we'll be back here next Wednesday, 11:00 a.m. Pacific, 2:00 p.m. Eastern time, 1900 UTC, to do the show all over again. Episode 389 will be, I guess, next time.

**Steve:** Yes, a Q&A.

**Leo:** A Q&A. Thank you, Steve. Thank you, everybody. We'll see you next week on Security Now!.

**Steve:** Thanks, Leo.

**Leo:** Bye bye.