
# Browser Password Managers

**Description:** This week Steve and Leo discuss the week's more interesting security news, including HP's recent analysis of the (lack of) security in "Internet of Things" appliances, and the forthcoming Black Hat presentation on "BadUSB" which generated a lot of overly hysterical press coverage. Then Steve summarizes his analysis of the Browser-based Password Manager research to be released later this month.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-467.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-467-lq.mp3

---

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. He does have some insight into the BadUSB exploit, ahead of the Black Hat reveal. He'll also talk about password managers and one thing nobody should try. It's all coming up next on Security Now!.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 467, recorded August 5th, 2014: Browser Password Managers. It's time for Security Now!, the show where we cover your security and privacy online with this fellow right here, Mr. Steven "Tiberius" Gibson, our security guru.

**Steve Gibson:** Yo, Leo.

**Leo:** GRC.com. Hi, Steve.

**Steve:** Hey.

**Leo:** Is this an unusually busy week for you? Because Black Hat is this week.

**Steve:** You know, I was thinking about that. Not only do we have Black Hat this week, but in a couple weeks we have the 23rd Annual USENIX Security Conference. I think it's down in San Diego. I'm not sure. And that's where the guys who are going to deliver this Browser Password Managers paper and research are presenting. So the week - I think it will be next week's podcast which is, I mean, I would imagine Black Hat is probably going to provide us with weeks, plural, weeks of interesting stuff. I scrolled through the listing

of presentations. It's like, ooh, that could be good. Ooh, that could be good. And, ooh, you know.

So, and of course today we're going to talk about BadUSB, which is being presented in two days, on Thursday. And the press just went ballistic, as they tend to. Uh, and so we have two major things I want to talk about, which is we are going to get to the Browser Password Managers page, which research will be delivered, as I mentioned, later this month. And we have to talk about BadUSB. HP did a quick little sort of self-serving analysis, but still interesting, analysis of the security of 10 Internet of Things gadgets. Uh, a 17-year-old Australian figured out how to disable PayPal's two-factor authentication. And I wanted to talk briefly, I got some numbers and things, about Google's identifying that photography, the...

**Leo:** Child porn image.

**Steve:** The child porn image in someone's email. So, and of course that sort of like raises people's hackles. It's like, wait a minute, Google's looking at our email? And I saw you on TWiT mention the good news that it's, I think, a hash, but we've got to...

**Leo:** I think it was kind of fun. We figured it out on TWiT. At first I was - my hackles were, you know, raised. And I brought it up because of that, like what does this mean? And fortunately we had a smart panel on Sunday, and they tossed me off the ledge. So I think it was Chad really gets credit for finding the article, and then Lindsey Turrentine explained how that might work. You know, what I didn't know, Allyn Malventano was on TWiT Sunday, as well, and he was an NSA analyst.

**Steve:** Yeah.

**Leo:** He never told us that. We knew he was a Navy chief and a submariner.

**Steve:** Yeah. It was fun, too, because he was on the down low. He said, "Well, I really can't talk too much more about that."

**Leo:** He made a big point of saying, "But look, we had rules, and we were trained carefully to follow those rules."

**Steve:** Yes. I thought those were good. You know, he made, you're right, he made a point of saying that, from the outside, the Snowden stuff seems extreme. But he was sitting at a terminal, and he was able to say, look, you have to understand, at every stage we were being told, if anything comes up, or you run across anything that deals with Americans, delete it, or change the page, or stop, or hit escape, or whatever it is. I mean, so it was really pounded into them that they are not here to dredge up information or have any contact with U.S. citizens. So it was good to hear from somebody who has got no cross to bear on the inside.

**Leo:** Yeah. I thought so. I thought so. So password managers. We're going to talk about BadUSB.

**Steve:** Yup.

**Leo:** I don't know if you want to talk about the Synology CryptoLocker variant. Have you heard about that?

**Steve:** Heard about it. Haven't had a chance to dig into it yet.

**Leo:** There's probably not much to say except that, if you have a Synology that is out on the public Internet, you want to make sure you use the latest version of their DSM software, the Synology Disk[Station] Manager, whatever it is, 5.0. Synology's analyzing it, and we'll know later in the day if it's susceptible or not. But they think it's not. What we did here, we have a few Synologies that are public so that we can use them. And we've taken them offline.

**Steve:** Oh, good. And so, yeah, so just for those who don't know, that's a Network Attached Storage system.

**Leo:** Oh, yes, thank you, yeah. Minor detail. What is that?

**Steve:** Yeah. And so apparently the CryptoLocker folks decided, hey, there's something huge we can encrypt. Lord knows what they charge in order to decrypt.

**Leo:** 0.6 bitcoins.

**Steve:** Okay. Well, that's, you know...

**Leo:** That's a little more than 350 bucks, I think.

**Steve:** Not pocket change.

**Leo:** No.

**Steve:** But, yeah.

**Leo:** And, well, here's what's interesting about it, and it doesn't bode well. Instead of being a pull attack, like you get email, and then you put malware on your system,

because the Synologies are online, this is a worm.

**Steve:** Yes.

**Leo:** It's a push attack. So that's a little worrisome.

**Steve:** Yes, that's big worrisome.

**Leo:** We'll keep an eye on that. Leo Laporte, Steve Gibson, talking about the tech security news of the day.

**Steve:** So HP made a little bit of news that got picked up in the press I thought was interesting, just because we've been talking about a huge problem that we're going to see in the future. And the fact that one of the reasons that this has occurred is that companies that are producing Internet-connected appliances haven't had, like, application-targeted protocols that they could use. Anybody doing a server, a router, a browser, there's a foundation of protocols that you just implement, and you have a solution. But we don't really have anything that fits the particular profile that Internet of Things devices require.

The good news is the industry is now awake to the issue of security and the fact that the Internet of Things is a thing. And as we talked about a couple podcasts ago, there are three different sort of interrelated efforts underway, specifically to create a protocol for light bulbs and dishwashers and the famous pasta machine and things that we want to connect for whatever reason to the Internet.

Anyway, HP has an application security unit called Fortify. And they did an analysis of the 10 most popular consumer Internet things on the market. And they didn't tell us which ones, actually for the sake of those things because that would probably be a problem. But they just put together sort of a nice sketch of what they found. They reported a total, within those 10 devices, 250 individual security vulnerabilities in aggregate for, like, an average of 25 faults each, although they weren't evenly distributed. So they didn't identify the specific things. But they did give us some bullet points like these are the kind of - this is what we consider a security problem.

So eight of the 10 devices failed to require a stronger password than 1234, so either on the device or on its corresponding support or connection or control site. So again, we're sort of seeing some of the same learning pains that we've all gone through over the last 15 years on the Internet being repeated. And device manufacturers are saying, well, you know, do we really need a strong password on a light bulb? You know, who cares?

And of course the problem is that, if that light bulb also has your WiFi password, which is strong, but its access password is weak, then potentially someone could get to it and then use it to bootstrap themselves into your greater network, which is why last week in fact one of our Q&A questions was how do I really create a segmented WiFi network where I could put all of those things I don't really - I'm not sure I can trust until they mature, off on their own space, and not risk my mature PC and Mac and mobile device network, where security has been more of a focus. Seven of the 10 devices did no encryption when communicating with the Internet or a local network. Six of the devices

had weak security on their interfaces; were vulnerable to persistent cross-site scripting attacks, which we've talked about in detail in the podcast; weak default sign-on credentials; sometimes didn't even encrypt their passwords. They just sent them in the clear.

So it's distressing to see in an environment where security is not hard anymore. It's not difficult. And you could argue that people are being trained to expect it, I mean, like, to be willing to come up with a longer password than 1234; that it's like we're starting over from scratch. However, I do think that, from the initiatives we've been seeing, this will get fixed quickly. We're not going to drag this on for 10 years. But our listeners need to be aware that these devices are not currently very mature.

Oh, and we've talked about this in various contexts. But six of the devices did not encrypt the software updates during download. So as we've seen, bad guys could create, and actually have created, malware which abuses the ability of them to essentially treat these like little computers that are also on your network. And nine of the 10 devices collected at least some kind of personal information - email address, home address, name, date of birth, whatever.

So anyway, that was HP's little, like, look, the security of these things is important. And Fortify is something that they're selling that's supposed to test the security of these. And I think this was HP's way of saying, "You know, folks, if you used our Fortify facility, you would have found these." I'm not sure whether the people who are making these things care yet. I think the market's going to have to make them shape up.

There was an interesting bypass, and not the first one, but this is actually more worrisome, of PayPal's two-factor authentication. We've talked about this years back. I still have, right here, still working, going strong, my little original PayPal football.

**Leo:** Me, too.

**Steve:** That we - yeah.

**Leo:** We talked about those a lot.

**Steve:** Oh, yeah. And I'm very conscious - I assume I can change the battery when it dies. I don't want to have to switch to the eInk card. But I turn it on, memorize the six digits, and turn it off immediately so that whatever battery power it has, it keeps as long as it can because I'm a believer in that. The bad news is what this Joshua Rogers, who's a 17-year-old white hat hacker in Australia - he contacted PayPal two months ago, and he said, you know, I really think you've made this, like, too easy.

What he discovered was - and it sounds like he was actually doing this himself. He has a PayPal account, and he was an eBay user. And some time ago PayPal allowed eBay, or eBay allowed PayPal, I'm not sure which direction it came from, to link your accounts. And I actually use it myself, and it's convenient when I - because I sometimes see old antique-y computers and things on PayPal, I mean on eBay, and I think, oh, I have to add that to my collection. So you're able to just say "Buy It." And once you've linked your eBay account to your PayPal account, that's all you need to do.

Now, you have to be logged in on both. So I think that's the way PayPal justifies the fact

that this is a second-factor bypass. I mean, any other time that I'm purchasing something with PayPal, I'm a big believer in restricting the spread of my personal credit card information, especially if it's a site that seems a little sketchy, where it's like, okay, I really don't want to give these people my credit card information. I'm happy when I see a PayPal button there. When I click it, invariably they bounce me to PayPal. I look at the URL, make sure that it's an EV cert so it's green and glowing. I'll float the cursor over, and it'll say "Identity verified by VeriSign" because that's where PayPal gets their certs.

So I do that stuff. And when I enter my username and password, it then sees that I'm set up for two-factor authentication, bounces me to there, where I have to press the button on my little football, get six digits, put it in and so forth. And all that works. None of that's necessary when I buy something through eBay. After I've linked my accounts, it's just done. And so, okay, how was that done?

Well, it appears that it was - and there's been no response. What happened was Joshua notified - he did the responsible disclosure thing. Two months ago he notified PayPal that just by putting the phrase "=_integrated-registration" in - adding that to a URL inbound to PayPal, they don't do second-factor authentication. It's unbelievable. And it's like, oh.

And in fact he has, on his blog page, I've got links in the show notes if anyone's interested, on his blog page he's got this big monster hairy link. But sure enough, you can click it, and it's got that little phrase added to the URL, and it doesn't ask you for a second factor. You do need the first factor, so you need to know the user's password. But the whole point of second-factor is that that may not be secure enough, so you need something that is, in the case of the football, is a time-based six digits that's changing every 30 seconds. So, but you say "=_integrated-registration"? No football. No second factor, even if you're using the eInk card. So Joshua, good going.

**Leo:** Now everybody knows how to do that. And I'm going to, in fact, I'm going to even use it.

**Steve:** Yeah, it's why I won't give you this factor.

**Leo:** Put my football away.

**Steve:** Don't have your football with you? Just add that to it. Yeah. And so...

**Leo:** I've always felt like places like that, where they have then a link, it says "If you don't have your football." And a lot of times they really bypass the second factor in general.

**Steve:** I know. I know. And in fact it's funny you mention that because I just launched SQRL. And if I look at - I get the exact - Settings and Options. I've got one here. "Request no recourse identity lock."

**Leo:** There you go.

**Steve:** And "Request disable non-SQRL login."

**Leo:** Love it.

**Steve:** And so the idea is, if you are - once you become confident of the way SQRL is working for you, you can turn these on. They get stored with your identity. And those are - there's no way at the client side to enforce it. But it indicates a desire on the user for the behavior of the server. And so it's - and that's why I phrase it as "request" because I want to make sure people didn't misunderstand...

**Leo:** It's optional.

**Steve:** Yes, yes. We're asking the site, please set a flag at your end that this is what I'm going to use because that's the way the user gets the maximum level of security. And so, exactly as you said, all of these things, which are "I don't have my football with me," well, okay. If there was an option I had to remove that from my account, I would, because I'll take - the whole point is I want to take responsibility. Otherwise you don't really have second-factor authentication. You've just got more, you know, a way of the human factor working around it. Exactly as you say, Leo.

So Google made some news that we were talking about because they spotted some illegal explicit child pornography in somebody's email, someone's Google Mail, and notified the authorities. So we learned a few things which were interesting. A Google spokesperson told Business Insider, that broke the story: "All Internet companies have to deal with child sexual abuse. It's why Google actively removes illegal imagery from our services including search and Gmail and immediately reports abuse to NCMEC," which is the National Center for Missing & Exploited Children."

**Leo:** Are they legally required to do that?

**Steve:** Well, the way - yes. The way it works is, if they know, they must report.

**Leo:** Got it.

**Steve:** So if they don't look, then they cannot be held responsible. But Google wants to. Jacquelline Fuller is the director of Google Giving. And she blogged about this, actually about a year ago, and had some good information there, too. She said: "In 2011, the National Center for Missing & Exploited Children's CyberTipline Child Victim Identification Program" - there's an acronym for you - "reviewed 17.3 million images and videos of suspected child sexual abuse. This is four times more than what their Exploited Children's Division saw in 2007." So that's, what, four years' difference. Factor of four growth of these images in four years. And she says, "And the number is still growing. Behind these images are real, vulnerable kids who are sexually victimized and victimized further through the distribution of their images."

Then she says: "Since 2008 we've used hashing technology to tag known child sexual abuse images, allowing us to identify duplicate images which may exist elsewhere. Each

offending image in effect gets a unique ID that our computers can recognize without humans having to view them again. Recently, we've started working to incorporate encrypted fingerprints of child sexual abuse images into a cross-industry database. This will enable companies, law enforcement, and charities to better collaborate on detecting and removing these images, and to take action against the criminals. Today we've also announced...." And it goes on. But so the idea is there is an industry-wide consortium who are finding these images, hashing them, and collectively creating a shared database of hashes. And if ISPs or email providers and so forth spot the hash collisions, then that causes them to look more closely.

Now, it's not clear to me how the images are first seen, who or where initially looks at these and decides, ooh, wow, that's not okay, and then submits it. Someone is doing that somewhere. We assume that Google is just using an automated hash collision in order to make that determination. So, but that's what the technology is. And so that's...

**Leo:** Yeah, and so people are very upset that Google is looking at their mail. But it strikes me, I mean, nobody, first of all, no human is looking at this. That's not - or maybe they are, but that's not necessary to match these hashes.

**Steve:** Correct.

**Leo:** These are generated. It's not a big deal to generate a hash across every image. You're already probably compressing it and doing other stuff to it anyway.

**Steve:** Right.

**Leo:** You're moving it around. You're storing it on servers. You could compute a hash quickly. If it matches the database, then I imagine something happens, whether it's a Google human or the National Center for Missing & Exploited Kids human. At that point some human probably looks at that image to make sure that the hash is not a mismatch, although as we talked about, hashing technology is very good. If you have a hash match, I would imagine that's the image; right?

**Steve:** Well, yes. I mean, we know that there's the possibility of a collision. We don't know how large the hashes are. If it was an SHA-256, that's what everybody is using now. It's what I'm using for SQRL IDs. It's what Bitcoin is using for user addresses. I mean, those are, you know, bits are cheap. And an SHA-256 just has a, I can't say zero, but it's so ridiculously close to zero. I think it's got, I've forgotten now how many, zero point - and it's like 77 zeroes or something, and then a two or something out at the end.

**Leo:** Of course, if they drew a moustache on the kid, then the hash would be broken.

**Steve:** Yes.

**Leo:** But that's not how these guys tend to work. They collect these...

**Steve:** Correct. They're not technical. Yes.

**Leo:** Well, even if they were...

**Steve:** All you would have to do is re- if you just recompressed it at a different JPEG - and actually, even if you just recompressed the JPEG, it would end up with a different hash.

**Leo:** Right, but that's not what they're doing. They're collecting. There are thousands of images. And so I think there's nothing wrong with this. This is great. It does not intrude on your privacy in any way.

**Steve:** Yes. Ultra-low false positive rate. And if it did collide, I'm sure it comes to the attention of a person.

**Leo:** Oh, a human looks at it at some point before...

**Steve:** And if it's a tree, then they're like, okay, fine, you know, he likes maples.

**Leo:** Right.

**Steve:** So he's not a problem.

**Leo:** I just don't see - people are acting as if there's somebody reading their mail and looking at their images. That's not what's happening.

**Steve:** No. And as we know, Google's model is to have machines looking for keywords in mail in order to show you ads that are relevant to the subject. That's what you get in return, I mean, that's what happens in return for the free email system with all of its many bells and whistles and features.

**Leo:** Right, right.

**Steve:** So, yeah. Big deal.

**Leo:** Big deal.

**Steve:** It's a computer. So, BadUSB. BadUSB, bad.

**Leo:** I can't wait. I really want to hear your take on this.

**Steve:** Okay. So in two days, two German researchers at the German Security Research Labs, Karsten Nohl and Jakob Lell, are going to, well, I would shock the world, except that the cat's out of the bag. And, I mean, oh, my god. The headlines have been ridiculous. Ars Technica: "BadUSB exploit makes devices turn evil." Wired: "Why the Security of USB Is Fundamentally Broken." Gizmodo: "USB Has a Fundamental Security Flaw That You Can't Detect." And then I think ExtremeTech was maybe the most extreme, extreme hype. They said: "Massive, undetectable security flaw found in USB: It's time to get your PS/2 keyboard out of the cupboard." Like, okay.

So here's - okay. And I listened to you guys on TWiT, and you gave this some good coverage. And really you got everything right. The surprise is that we don't have numbers yet. I think this is going to - this is the kind of story that's going to end up having some legs to it because this is big, potentially. But they discovered that a surprising number of USB - and they're specifically addressing thumb drives, but it's probably bigger than that - are, in the first place, firmware-based. That is, USB is a sophisticated protocol. And so you don't just have a memory chip and hook it up to those four wires of USB. Actually two of them are 5V in ground power. So one wire is in serial data and the other is out.

But there's a very complicated protocol for which you need a processor behind it in order to exchange handshakes. For example, we've talked about TCP protocol, where you send a packet, and you get a packet from them, and you're negotiating things, and you're saying, hey, this is what I can do, and what can you do, and this back and forth. That requires intelligence at each end of the link. And over time that intelligence has become much cheaper and ubiquitous.

So what we've ended up with is computers, true computers to support this protocol in everything. If it's USB, it's a computer. Unless, and we were talking about this, too, technically a power outlet, a power adapter doesn't have to support the protocol. It's able to just say, you know, it just does the 5V and ground pins, and something hooks up and realizes, okay, this is just a power tap There's no brains here. And in fact it would be an evil power tap that did have brains because then it could try to get up to some nonsense. And that's an example of this kind of exploit.

But what these guys realized was, or basically have been the first to report, these computers unfortunately are reprogrammable. That is, they're non-volatile memory firmware. And the firmware can be changed. So they're not burned in ROM. And in fact they didn't even have a fusible link blown after they were written because there are a lot of technologies now where you can write something, and after you've written it and reread it and confirmed that it works, you blow a fuse in the chip that prevents that data from ever being read out. That's often done for proprietary reasons, to keep proprietary code in the chip in there. And then it will only execute the code. It won't allow you external access to it.

These guys have found - and we don't know details because we won't know for two days. We'll probably come back to this next week and just talk about what we actually learned from their report because everything that - I looked at their own site. They have a site, SRLabs.de/badusb. But there's not much more there yet. And you can't blame them. They want us all to care about what their presentation's going to be in two days. What

we know is that they spent apparently a substantial amount of time reverse-engineering some number of thumb drives.

And what their point is, is that USB is a powerful protocol. And they're going to show proof-of-concept exploits where what looks like, and was originally when it shipped, a 16GB thumb drive is now something else. They reprogrammed the firmware. And remember, it's a computer. We think of it as storage. But in order to support the USB protocol, you've got to have a computer. And so they've reprogrammed it to do bad, to be evil. And there are many things it could do.

For example, if somebody looks at it from its file system, it could look empty, except it could also, if it got left in the socket of a computer which is booted, it could see that, when power comes up, there's nobody around at that point. The OS is not yet booted. But we know that many systems will boot from USB. So it could suddenly offer a boot image at that time which is otherwise stealth and doesn't appear. So when you stick it in and look at it, it's empty. But when it's already in, and the computer powers up, it's a boot image, and it takes over.

So that's an example of what happens when you merge the intelligence that these devices all have with the ability to change what they do. It is a powerful protocol. Now, the reason this is not as dire, at least as the articles have said, and so far they've not talked about mitigation, that is, publicly, except to say, well, firmware images need to be signed. Okay, that won't work. I mean, it's not clear to me what signing the firmware image, how that helps. That might help maybe the device to defend itself against modification, but that's trivial for it to do. Just arrange it not to be writeable after you ship it. I mean, like with the fusible link or any of a number of approaches.

So it seems to me the right way to look at this is that the user has the onus, essentially, of responsibility. And that is what these guys are saying. They're saying, I mean, the hyperbole in these news stories is crazy. Like once a USB drive has ever been inserted into an untrusted computer, you can never trust it again. It's like, okay.

**Leo:** Well, a couple of questions come up, though. First of all...

**Steve:** Okay, good.

**Leo:** Do they have to make USB drives with writeable, electronically writeable firmware?

**Steve:** No.

**Leo:** You could make it with ROM.

**Steve:** Absolutely. Nothing - this must just be convenience.

**Leo:** It's convenience because they, in the manufacture process, may realize, oh, shoot, we screwed it up. Bring them all back, we'll rewrite the firmware.

**Steve:** Well, and they may well, for example, for efficiency, I would imagine that same processor is doing the memory management. We've talked a lot about like wear leveling and so forth. It may also be the memory controller. So in that case you just want to use one chip for all of your different memory types and sizes and configurations. So the programmability of it would be a huge cost saving. They can order 10 times more of one chip and then program it for the particular application that they're using it for.

**Leo:** Ah, of course. So there's a - okay. So - because I thought, hey, ROM has got to be cheaper. If you could get a perfect ROM, then that would be the cheapest way. But you're right. If you want to have a million chips and use half of them for one thing and half of them for another, then rewriteable firmware is good. But you could have it be PROM; right?

**Steve:** Yes, yes.

**Leo:** Doesn't have to be EEPROM.

**Steve:** Right. It ought to be - so, again, they don't care. It doesn't really matter to them. They've shipped you a thing that does what they said it does. It stores data. They also shipped you a reprogrammable computer that you weren't expecting. And so it doesn't hurt them. I mean, they have no obligation. Now, maybe if this gets enough traction they'll start advertising, you know, "Locked-down thumb drive, cannot be reprogrammed." You know, "Guaranteed not to be evil." Who knows.

**Leo:** Oh, I bet somebody will. And that's the other question is, is anybody doing that now, just because they do? We said on TWiT, and I don't know why, who said this. I can't remember if it was Allyn or maybe Lindsey, that IronKey, which of course we've talked about as the ultimate in secure USB storage, is vulnerable. I don't know if it is.

**Steve:** Well, it's explicitly updateable. The firmware can be updated. And we don't know...

**Leo:** And they make that a selling point.

**Steve:** Yup.

**Leo:** Oh, boy.

**Steve:** Now, okay. So the solution: USB is divided into device classes. And we all know what they are: audio, I/O, modem, Ethernet, WiFi, keyboard, mouse, joystick, video, webcam, scanner, printer, mass storage. It could be a hub, a Bluetooth adapter, an IR. I mean, think about it. Think about how incredibly versatile this thing is. I mean, this has been an incredible success. And I saw some stories talking about this was being the fault of the USB Consortium not providing security. It's like, wait a minute. Let's remember

our history.

All of this predated any concern for history. I mean, people were using "monkey" as their approved password back when USB was starting. The fact that this worked at all was a miracle, let alone worrying about the security of it. So what we have is a very versatile, very powerful bus. But unlike Firewire and Thunderbolt, which we've talked about, which are - they are bus mastering-capable, meaning that, if you have an active Firewire port or Thunderbolt, which is the same technology, you actually can get on the bus, that is, the system's processor bus. You can suck memory, like the main memory out of the laptop by having a Firewire port.

**Leo:** DMA. Direct Memory Access.

**Steve:** Yes, exactly, Direct Memory Access. USB is not that. USB is a host-managed master/slave relationship. So when you plug a USB device into your computer, and this protocol I've been talking about starts up, in this interchange the computer says, the host says to the slave, or the master says to the slave, "What are you?" And the thing says, "I'm a speaker." And so the OS says, okay, looks around to see if it's got speaker drivers for the USB, and says, "Oh, yeah, okay. We're set to go." And it loads those up, and off we go. Or it says, "I'm a keyboard." And the computer says, ah, okay. And so there's an HID, a Human Interface Device driver which it connects up and becomes a keyboard.

The point is this is completely under the management of the operating system. Today, because this has never been a concern, that is, when you plug in a thumb drive, it's a thumb drive. And it identifies it as such. If you plugged it in and it said it was a keyboard - and that's one of the exploits these guys talk about. Remember that when I encountered Stina at the top of the escalator at the security conference in San Francisco, and she said, "I have a one-time password device," and it was little tiny USB thing. And I said [gasps]. Because I immediately understood that this would type this crypto thing in for you. It was the power of USB that made that possible.

**Leo:** Yeah. But now that I think of it, YubiKey is subject to BadUSB, too, isn't it. You could reprogram a YubiKey. It's writeable.

**Steve:** There is some control. I don't know how much they offer.

**Leo:** May not have a lot of storage.

**Steve:** Yeah.

**Leo:** But you could reprogram it.

**Steve:** So here's my point. We don't know, I mean, this has got people worried. And what these guys have discovered, or really just noticed, is that because these devices are now reprogrammable computers, and they apparently can be taken over, you don't have to break them open and put wires on them. Apparently they can be reprogrammed

through the USB, that is, there is a way to get into the device just through the USB. That would allow malware to jump onto the device when you plug it into someone's computer. Now, again, this is, you know, it's like what version, what type of processor, is it a custom chip, I mean, there's like a whole bunch of other things that all have to line up perfectly for this to work. But the possibility is there.

**Leo:** At the Black Hat they're going to demonstrate it with a particular chip from a particular company.

**Steve:** Right. But if you plug this into your computer, right now, because we're all trusting USB, the so-called "enumeration" occurs automatically. All we have to do, if the industry decides this is a problem, is unfortunately there's going to be some user cost, that is, in terms of involvement. Remember the convenience versus security. But it would pop up a dialogue and say, "Hi there. This thing is trying to be a keyboard." And you say, wait a minute. That's storage. It shouldn't be able to type anything. And so you say, "No, don't let this be a keyboard."

So my point is the master, the host, has ultimate say, ultimate control over whatever it is you plug into your USB port. The USB device declares what it is. At the moment, for ease of use and because we've all been trusting of USB, the OS just does it for us. All we have to do is bring up a dialogue and give permission, look at what this thing is asking for, very much like when we install an app on our phone. It says, "This app wants access to the following resources. Do you want to give it to it?" And similarly, it would be this USB thing wants to be able to do the following things on your computer.

**Leo:** But that wouldn't mitigate all the potential attack vectors. For instance, they describe how you could download onto a USB key a Ubuntu installation, verify the MD5. But between the time that you downloaded and put it on the USB key, and the time you used you used it to install Ubuntu, it could be compromised by that USB.

**Steve:** Yes.

**Leo:** So that wouldn't mitigate that. You're using it as mass storage at every point.

**Steve:** Good point. Good point. Yes. Yes. You've got, I mean, fundamentally we're plugging an untrusted computer into our computer.

**Leo:** It would eliminate a lot of the attacks because they did talk about some of the attacks would turn a USB key, a drive into a keyboard and that kind of thing.

**Steve:** Right.

**Leo:** But it wouldn't be a hundred percent. I think, though, it's very important, a lot of people seem to misunderstand why this is an issue. Anything that you plug into your system, any manufac- you're trusting the manufacturer. Whether it's software,

hardware, whatever, you're always trusting the manufacturer. Unless you have a process of vetting the source code and looking at all the firmware, when you buy a PC from Dell, you're trusting Dell. This is a serious problem, not because the manufacturer could do it, but because it could be reprogrammed in the field by a bad actor later. Right?

**Steve:** Right. Right.

**Leo:** And, now, I don't know, do we know how easy this is to implement? Sounds like you need some specific hardware to do it.

**Steve:** We don't yet. And we'll find out. It is not clear whether all of these devices can be reprogrammed through the USB channel. I mean, from a manufacturing standpoint, it seems like it would be nice if it could. That is, if there was like some way of doing something to the signals when you powered up so that it didn't come up as a USB, but it kind of like came up in programmable ready mode or something. We just don't know.

I mean, the chips themselves typically have a huge number of pins, you know, 48 pins as opposed to the four pins on the USB bus. So, and there's a universal standard called JTAG which is a serial interface which probably all of these things support. I would imagine they used a JTAG interface, like probing around, finding that on the processor, maybe even identified the make and model of the chip. And once you have that, you go to the specs, and you know all about it. So it's not clear even how much reverse-engineering they had to do. We've heard stories, for example, of people changing the firmware on hard drives.

**Leo:** Right.

**Steve:** So hard drives are the same deal.

**Leo:** Now, I think hard drive manufacturers since then have started locking their firmware down.

**Steve:** Right.

**Leo:** That's my vague memory. Is that right?

**Steve:** And it certainly makes sense, it certainly makes sense for a device like that, where you've got someone - first of all, not that many manufacturers. You've got Seagate. You've got Western Digital and so forth. They're not going to want the reputation cost of having hackable hard drives. And they've also got a ton of proprietary interest in the technology that they're not wanting to leak out. There's really nothing proprietary about a thumb drive. I mean, they're a commodity. They're in fish bowls at the checkout stand. So it's like...

**Leo:** It's also probably the case that this was not a great discovery, but just a light bulb went off for somebody because we've kind of known that these reprogrammable chips are in them all along. Right?

**Steve:** Well, and you wonder how long the NSA has known.

**Leo:** The NSA has clearly known it for a while.

**Steve:** Uh-huh.

**Leo:** So, I mean, but this is - it's not like a major - it's not really a discovery. It's like a, oh, you know what?

**Steve:** Right, right.

**Leo:** That could be bad.

**Steve:** Yes, it's like in order to do USB, you have to have a processor. And if that processor is accessible, and if you're clever, exactly as you said, Leo, changing the image on the fly. I mean, say that you sold it as a 32GB USB, and it was actually 64. Now it's got a whole archive. I mean, it's got every possible version of Linux that has ever existed, like sitting in the back room, able to roll it out whenever it wants to.

**Leo:** Right. Then another thing we don't know, and I'm sure we'll learn, is if you have to make these rewriteable, or if you could make ROM-based firmware USB drives that just couldn't be modified.

**Steve:** Yeah. From a technical standpoint, the firmware is logically separate from the data. The firmware...

**Leo:** Right. It's not on the drive. It's in a chip.

**Steve:** Yes, yes. And so it just - I see nothing at all except they didn't care.

**Leo:** Convenience.

**Steve:** When there are $5 thumb drives at the checkout stand coming in bulk quantity from China, who cares? They just - it's whatever's cheapest for them, that's what they'll do. But, yeah, as this becomes, I guess - so the immediate takeaway is the point these guys have raised is valid. And that is, every single place you stick your USB, you need to make sure, you need to be sure it's safe because - and again, there are so many

different chips.

First of all, we don't - it's not really clear how possible it is to upload firmware through the USB interface. I know that what you're seeing is USB storage. And we'll have to see how possible it is to actually reprogram the processor through the USB bus because that's different than cracking it apart, getting access to the chip's pins themselves, and creating an evil USB - which, for example, the NSA could do. So it's very different than to plug it in and have it infected through the USB channel. That makes the threat much worse, but also I think it's less likely, it's less likely that that can even be done.

**Leo:** Another point, we know that Stuxnet was spread to the Iranian centrifuges in their uranium enrichment plant through USB keys. But it was not likely this technique; right? It was probably autorun stuff on it. Or was it? Do we know?

**Steve:** We're thinking that it was unpatched machines that they were counting on - they knew that these machines were not networked, that they were air gapped on purpose, but that it was necessary to move data between these separate networks, and that they were able to use flaws in the USB protocol on the target operating systems. That's what we believe...

**Leo:** Different technique, then.

**Steve:** Well, we don't know for sure.

**Leo:** We don't know.

**Steve:** I mean, this stuff's been around for a long time.

**Leo:** Really interesting.

**Steve:** Yeah. So just wanted to make a note. This is not - this means nothing except it's a milestone for me and for the people in GRC's newsgroup. But yesterday I turned over the first pre-completion SQRL code. They are now testing stuff. But it's not done. Because where I got was self-contained, I was able to let people start playing. I have yet to do the identity import and export and backup, and the protocol. Even though they sound like a big deal, they're like, I've done the bulk of the work. And we immediately hit a problem with Aero. I don't have it. I mean, even on my Win7 machines, the first thing I do is turn that ridiculous cycle waste off. But there are a number of people who do have bleed-through faded borders on their windows. And it's like, it took me 10 minutes, and I fixed it. So that's working. Wine has a font-sizing problem I haven't even looked at. I'll look at it this afternoon. But I just wanted to say that another milestone is reached. The gang is now pounding on the UI. While they're doing that, I will work on finishing it. So more progress.

For last week's Q&A I ran across a question that actually tickled an old memory of mine because someone wrote and just said, "Steve, all of a sudden my external hard drive was not being recognized by Windows 7. The error I'm getting when it's plugged in using an

USB 2.0 SATA/IDE combo adapter is 'Unallocated 3.86GB unknown, not initialized,'" and he says, "though the drive is 2TB." He says, "If I try to initialize the disk drive I get an error, 'error (cyclic redundancy check).'" And he said, "My question is, if I purchase SpinRite, is it still possible to recover data from a dying disk drive? I love your show and have been listening for over a year now. I've learned a lot."

The reason that this kind of hit me as funny is I remember 20 years ago, when I had a room of tech support people at Gibson Research, in the early days of SpinRite, SpinRite 1 and 2, selling them on - we had both the 5.25 and 3.5" floppy drives and a manual and literature and things in the boxed product. And they were at Egghead and Microcenter and Fry's and so forth, back in the day. And I remember, if I would, like, need to talk to one of my tech support guys, I'd go to his desk, and the phone just rang, and he'd pick it up, and he said, "Gibson Research Tech Support," and then there'd be a long pause while he's listening to the person. He'd go, "Yes, it does. Okay, fine. I'll transfer you to sales." And he'd hang up. And one of these guys was named Karl. And I said, "Karl, what was that?" And he says, "Oh, that's what most of these calls are." I say, "What?" And he says, "They ask does it really work." [Laughter] Or back then it was, remember, it was also doing a nondestructive re-low-level format, very aggressive.

**Leo:** Yes, yes, yes.

**Steve:** So if people said, "Is this safe to use?"

**Leo:** "Yes, it is."

**Steve:** And, "Yes, it is."

**Leo:** "I'll transfer you to sales."

**Steve:** "I'll transfer you to sales." And then I was thinking, you know, I must have been affected by that because you'll remember my very short little slogan on the SpinRite pages at GRC, where I've got the big SpinRite logo, and just two little words down below. It says: "It works."

**Leo:** It works.

**Steve:** Anyway, so I don't remember whose - I don't know this guy's name who asked the question. I did respond to him. In fact, I wrote back, and I said, "I can't really say for sure, but your mention of CRC error is precisely the sort of trouble SpinRite was designed to find and fix. So I'd say that the chances were good. And if not, we'll be happy to refund your purchase price. Just tell sales that it didn't do what you hoped and needed." So...

**Leo:** That's nice. That's your [indiscernible].

**Steve:** Yeah, we do. We've always offered anybody who's not happy their money back because I would never want to have them keep it if they didn't like it because it didn't work.

**Leo:** That's nice of you. All right. Let's - it's time to move on.

**Steve:** Okay. So at the forthcoming USENIX 23rd Annual Security Symposium at 2:00 p.m. on Thursday, August 21st, four UC Berkeley researchers will present the result of an extremely, I would say an amazingly detailed deconstruction of five password managers, most of which we've heard of: LastPass, RoboForm, My1login, PasswordBox, and NeedMyPassword. Actually I don't think I've ever heard of NeedMyPassword before. And in fact it was a little disturbing. I had made a note here saying no auto-login for NeedMyPassword, no credential sharing, no password generation. The credential storage is only on their website. And the user's login credentials are not encrypted before being sent to NeedMyPassword.

**Leo:** See, that sounds bad in a lot of respects.

**Steve:** So this is, like, NSANeedsYourPassword is probably the right name for this, or maybe NSAHasYourPassword. So, yeah, these guys don't - they didn't really do it. So these four researchers went to amazing level to reverse-engineer, and from an adversarial standpoint. The first page of my show notes, Leo, has a diagram of one phase of the process they reverse-engineered from LastPass's generation of something. You know, one of the many diagrams. This is a 15-page detailed PDF. And I have the takeaway from it. I just want to see if there's anything - there were some other weird things they found.

For example, there's one called My1login. And they were a little concerned, and in their paper they said they would talk about this later, but they never did. Although they also have a more detailed technical report that they'll be producing later, not this, you know, so detailed as this 15 pages is, this is the short version of their research. And remember that I mentioned to you weeks ago that I was sure I'd seen somewhere that they were doing this before developing their own. And they do say that. I reencountered that at the end of this paper. So this is where it is. They're planning to implement their own password manager.

So I think that, assuming everything is just as they say, they just wanted to really understand the challenge of doing this securely, and it is a challenge, prior to doing their own. Anyway, this My1login has the weirdest hash. They take the user's password, even and odd characters separately, and then MD5 hash them.

**Leo:** What?

**Steve:** It's like, exactly.

**Leo:** Why?

**Steve:** Exactly. That I, like, a worry because it's as if somebody - first of all, you really don't want to use MD5 any longer because it just doesn't produce a large enough hash, so collisions or brute-forcing is not that difficult. In fact, maybe the reason they did even and odd is specifically to avoid password lookup because MD5 is small enough that maybe they could do - there are databases against commonly used passwords. Although all we would have to do is reverse them or something. Anyway, it sort of raised a red flag that they were using MD5. And then they were like, as if they thought that separating them in odd and even character groups and separately hashing them and then merging that would generate a better result. Or maybe they only had access to MD5 hash in JavaScript. It's hard to understand what they were thinking. But it's a little strange.

However, they're the only password manager that offers bookmarklets, which we'll talk about in a second, in a secure fashion, which I thought was interesting. So I don't mean to dump on these guys, and these researchers didn't either, because bookmarklets are extremely challenging to do in a secure fashion. So these guys looked at bookmarklet vulnerabilities. And of course the idea for a bookmarklet is that it's actually possible - we're familiar with bookmarks, which are basically a URL that you click on, and it takes you to a page. And it turns out that you can put JavaScript in a bookmark, and it becomes a "bookmarklet," as it's called, and it will execute. So this is convenient for mobile browsers that typically don't support plugins.

So, for example, with Chrome and Firefox, both that have mature, good plugin technologies, you don't need this because you install LastPass or RoboForm or whatever as an add-in to the browser, essentially enhancing the browser's intrinsic operation. But in Safari, on iOS, or browsers on Android, where you don't have a plugin architecture, yet you'd still like access to your cloud-synced usernames and passwords through a password manager, you need to run code. And bookmarklets are the way that's done.

The problem is that, and this is a big problem for the whole bookmarklet technology, when you click on the bookmarklet, it's running that JavaScript code, the bookmark's JavaScript code, in the context of the site that you're going to. So JavaScript needs to run in a web browser context. And the bookmark is not referring to a given site. It's essentially a way of injecting JavaScript into the current page. The problem is you are vulnerable to deliberately evil JavaScript on that page.

And normally you're in an iframe. Hopefully you're in an iframe. It wasn't really clear from their paper whether everybody who is supporting the bookmarklet option was using an iframe because browsers have gotten very good about supporting the separation of the context within an iframe from what's on the page. However, these guys take the position, and from what they said I got the sense that perhaps not all of the password managers are using iframes with bookmarklets, which makes them a concern.

The problem is that, if you click on the bookmarklet to log into a site, and you're not yet logged into your password manager, the only safe thing to do is leave that tab and go open a new tab and log into the password manager, then come back. Only My1login does that. And so even though in some ways they're a little unsophisticated, it may have been in fact that they don't have the technology to allow you to do an in-place login, it turns out that's the only thing these researchers believe is safe, the reason being that it's inherently problematical to run any JavaScript in the context of another page which itself has to have JavaScript enabled. In order for your bookmarklet JavaScript to run, that page has got to be running JavaScript. And in order to create the iframe, JavaScript has to run outside the iframe in order to instantiate the iframe, which means there is an opportunity to subvert that.

So what this ultimately means is, if you were not already logged into your password

manager, and you were using a bookmarklet to log into a site which was evil and had sophisticated scripting to recognize a number of different password managers, you would be presented with a dialogue on the page created by the bookmarklet, telling you you were not logged on to your password manager, and asking you to do it right there. And these guys make what I think is a very good point, which is this is training people to put their password manager credentials in when the URL says www.youbetternottrustthissite.com because that's the nature of an iframe is that you're not seeing the URL of the iframe. That's hidden. You're seeing the URL of the page that's hosting the iframe. And if there's scripting running on the page, it could have subverted the bookmarklet's code before the iframe got created. And what that means is a 100% full credential capture. They would capture, this evil site could capture your password manager login credentials. And then they would have all of your credentials managed by that. So obviously that's not good.

So the main takeaway from this paper in terms of, like, what did we learn from it, what can you do, is if you're a user of bookmarklets, log into the password manager on a tab on that manager's site. Go there. Log in. Your browser will then have the authentication cookie for that domain. Then when you use the bookmarklet - and, I mean, I got a sense their position is, if web developers are really, really careful, you can probably use bookmarklets securely.

But I came away with more of a feeling of concern than comfort from this. It's just sort of a last-ditch measure. Well, for example, I don't use them on iOS. When I run across, when I'm logging into something on iOS, I will go over to my LastPass tab browser, open - and if I'm not using that browser, I just generally use Safari, I will open the vault, and I copy my password first onto the clipboard, then move over to go back to Safari, paste that in. Then I go back over and get my username, and I do it second because that way it overrides the password on the clipboard, and I'm not leaving it there.

**Leo:** Oh, that's smart. You're smart.

**Steve:** And then I move back over, put my username in, and log in. I mean, yes, it's not as convenient as clicking. But just, to me, I understand that better. The problem is there's just, if you don't have a plugin, you really don't have good security containment. And running JavaScript in the context of a site that you can't vouch for is just - it is worrisome. And that's the way bookmarklets operate.

**Leo:** It's going to get better with iOS 8 and the next iPhone because extensions will allow, 1Password's already said they're going to do this, and I presume LastPass will, too, kind of interact with Safari and do it as they do now on Android, just do it as you're there.

**Steve:** Yes. And speaking of iOS, we should note the news that 8 is scaling back those security concerns that Jonathan pointed out and we've talked about for two weeks. Like the pcap daemon is no longer running in the background all the time.

**Leo:** This is in Beta 5 of iOS 8. They took out a lot of the stuff people were bugged about. So that's good.

**Steve:** And they specific wording, I had to - it was just happening as I was getting ready for the podcast. But I saw some specific wording that said those will no longer be available over wireless protocols. So it sounds like they're deliberately saying, "Oh, you know, you're right, this is too permissive. We need to back down on that."

**Leo:** They're responding, yeah.

**Steve:** Yeah, that's great.

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

**Steve:** And very quickly, yeah. And better in 8 than in 9. I mean, I'm glad that we are at this place in the cycle where they're able to do that and work it out. Okay, so...

**Leo:** LastPass. What do you think?

**Steve:** The bottom line was, remember that - I don't think I did say this, or not this week. This was all done a year ago. This was last August that the research was performed. All of the vendors were notified of this four-man team's findings. And all but NeedMyPassword responded and fixed whatever was known within days. So...

**Leo:** Oh, so this has been mitigated almost a year ago.

**Steve:** Yes. And what they found was instructive because it helps to demonstrate just how hard this is to do. There are two known classes of problems, sort of general web vulnerabilities, which we've talked about. We did separate podcasts on them each. One is a CSRF, Cross-Site Request Forgery. And that exploits the trust a site has in a user's browser. So the user's browser has authentication that causes a website to trust it. But a cross-site request forgery is a way of an attacker essentially masquerading as that user's browser's credentials, essentially getting that, abusing the site's trust in the browser. The other is cross-site scripting, where that exploits the trust a user has in the site. So like where you believe you're at one site, and you're actually at a different site.

So as an example, they found an attack on LastPass which I can live with, even if it wasn't fixed a year ago. And this gives you a sense for how hard these guys had to work, even a year ago, to find anything. A LastPass user must have created one of those LastPass one-time passwords, you know, those long strings you're able to just ask LastPass to give you some. And you can, like, have them in your wallet, or put them somewhere as an emergency authentication. If the attacker knows the user's username, so they only need the username, and somehow arranges to run their code inside the user's browser, so this is a cross-site request forgery attack.

And in the paper they don't explain how that happens, so there's another barrier. You know, somehow you've got to go to a malicious site which is running - and then it's running JavaScript in your browser. You have created a one-time password, and they know your LastPass username. Then even though they don't know that one-time password, they're able to, running in your browser and essentially trading on your

credentials with LastPass, obtain a copy of the encrypted credential database. Now, they have no way of knowing your one-time password, so they cannot use it to - they can't use what they don't know to decrypt the database.

And so the only thing these guys were able to do is a little more than nothing, but there were three things. They said the database is encrypted, but the website names are not. So the attacker can know the sites the user uses with LastPass. And remember, all these other things have to happen first in order for them to get the encrypted blob. And who knows. And again, this is a year ago. So maybe Joe is now encrypting the domain names, if that's possible. I don't know.

Armed with the encrypted record, an attacker could brute-force the user's master LastPass password. But remember, that's now PBKDF2, like, a thousand times. So there's password-based key derivation function, making that completely infeasible unless they use a really dumb password. And LastPass now chides you for trying to use really dumb passwords, so even that's not easy any longer. And it, like, audits your passwords to make sure that they're not bad. So all kinds of mitigations are up against that.

And then finally they said the attacker, impersonating the user through cross-site request forgery, can delete credentials from the LastPass database, despite being unable to obtain them. So your security is preserved, but someone could get up to some mischief a year ago if they were able to get themselves into this position all only if you had unused one-time passwords as sort of a gating factor. Because, again, the guys at LastPass worked, tried to work through giving people options and making them secure; but, as is always the case, with those additional aspects or factors or convenience comes some tradeoff. Sounds like they did a good job in making it as difficult as possible for anyone to get any leverage.

And by comparison, again a year ago, using cross-site request forgery in RoboForm, it is possible for an attacker to update, delete, or add arbitrary credentials to a user's RoboForm credentials database. It's like, okay. Again, not decrypt them, not impersonate the user, not do anything they really want to, but just sort of mess with them.

And then there is a cross-site scripting exploit in NeedMyPassword which allows for a complete account takeover. So that one was the scariest one. They didn't give us any details. And I would say, well, but it was a year ago, except that the NeedMyPassword people never responded. They're the one of the five that never responded in any way. Maybe they fixed them. Maybe they never even got the message. Who knows?

And then, lastly - those were under the category of web vulnerabilities. The final was, again, a really sort of edge case authorization vulnerability. There are a number of these password managers which allow credential sharing, where you're able to authorize somebody else who's also a member of the same password manager, who has an account with the same password manager. You're able to authorize them to use your login.

**Leo:** Right. I've done that to Lisa. I've sent my Wall Street Journal login to her via LastPass.

**Steve:** Exactly. Exactly. And that's tricky. It's another example of the kind of thing we would like to be able to do, but by having this be in the browser, it's challenging to do it safely. They gave examples where PasswordBox and My1login could be exploited given some really nearly impossible-to-create circumstances. In one example, My1login's

credential sharing gets authentication and authorization slightly confused, authenticating being obviously identifying who a user is, authorization being what that identified user is able to do.

And so these guys spotted a glitch in the logic. And the way this worked was, if the first user who had the credentials shared credentials with two others and revoked the credentials from one of the others, because there was a serial number that was trying to be unique, but simply incrementing - and unfortunately they didn't encipher it in order to make it an unpredictable value. And actually you could argue everyone would have to know what the cipher key was, so that wouldn't work either. So maybe they could have used a pseudorandom number, a big pseudorandom number. They didn't do that. They used an incrementing value. And then that allowed the other person, who was sharing credentials with the first, to collude with the first of the other people whose credentials had been revoked, and there was a chance that that person could still get credentials shared. It's like, whoa, okay.

I mean, so these guys were really pushing the limit of finding vulnerabilities. Again, this is good. This is what we need security researchers to do. They reported their results to these password managers, who fixed them a year ago and thanked them, I'm sure. So there wasn't anything that I saw as being, like, some disaster. I think anyone coming away from this in a couple weeks, when this is presented at USENIX, will think, okay, yes, this is not easy. We are trying to run trusted code in a fundamentally hostile environment.

Bookmarklets is maybe a bridge too far. Bookmarklets to me just seem like, eh, you're asking for trouble. You're asking for features. So there isn't another way to do this. You'd absolutely - the key takeaway, if you're a bookmarklet user, do not ever log into your password manager on the page that you're trying to log into. Go to a different tab, log in, come back, so that you're already logged in when you use the bookmarklet. That I would say.

But again, remember, for even that to be vulnerable, there would have to be JavaScript on that page explicitly waiting, I mean, just laying in wait for someone to use bookmarklets that they have reverse-engineered and are going to abuse. So while it's possible, it's not the case that Amazon is going to do that. When you've received a page over an HTTPS connection, you trust the JavaScript, and you're wanting to just use the bookmarklet to log you into Amazon because you no longer remember any of your passwords. And of course, with any luck, we'll have SQRL that will save us from all of this before long.

**Leo:** It all comes back to SQRL, doesn't it. It always comes down to SQRL.

**Steve:** Bushytailed little devil.

**Leo:** You still use LastPass. I still use LastPass.

**Steve:** Yup, yup.

**Leo:** Until somebody gives me a reason not to, I shall continue.

**Steve:** Exactly. I will, with any luck, there will be a phase-over to SQRL in the future. But it'll be LastPass until then.

**Leo:** Oh, I guess SQRL, well, SQRL would replace LastPass in terms of login passwords. But I use LastPass to store my Social Security number, things like that.

**Steve:** Yes, yes. I actually do, too. I use it as my form fill-in a lot, also.

**Leo:** And for credit cards, yes.

**Steve:** Yes.

**Leo:** Very handy.

**Steve:** And I trust it for that.

**Leo:** Yeah. You can do that in the browser, but why?

**Steve:** Yeah.

**Leo:** Steven, we've come to the end of another fabulous edition of Security Now!. Every week. I tried to watch "The Strain." It's too spooky for me, by the way. Are you still enjoying it?

**Steve:** Yeah, I'm a few episodes behind because I've just got, you know, Sunday night has become...

**Leo:** Too much. It's crazy.

**Steve:** Oh my goodness, yeah. "Masters of Sex" is back. Are you watching that?

**Leo:** Oh, yeah.

**Steve:** Yeah. And...

**Leo:** Actually, I shouldn't do that because it's not - it's a sexy show, but it's not about sex.

**Steve:** No. Yes. No.

**Leo:** It's about Masters and Johnson. And it's really about - it's more and more about women and their sexuality in the '50s. And not just women. Gays and all sorts of things.

**Steve:** There is a lot of that, yeah. It's just a great...

**Leo:** Yeah, it's really good.

**Steve:** It's a great HBO - no, it's not HBO.

**Leo:** Last week's episode, not this last Sunday, but the one before, "Fight," where it's the background is the boxing match? What a - one of the best episodes of TV I've ever seen.

**Steve:** Actually, that's exactly what Jenny said. She said she thinks it was, like, the best thing. She, like...

**Leo:** Mind-blowing.

**Steve:** Yes.

**Leo:** It's like a Broadway play.

**Steve:** Stunning, yes, that's exactly what she said, yes.

**Leo:** Yeah. The dialogue, everything, just really amazing. Um, yeah, so I watch that, too, yeah. Lisa doesn't like that. She doesn't - she's kind of sex negative. She doesn't - no, that's not true, either. Let's stop now before I get into too much trouble. The show is brought to you - she's got something she can throw at me, her laptop. I only said that because she was in the room.

**Steve:** Okay.

**Leo:** Yeah. This show is brought to you by the grace and good wit and charm of Mr. Steven Gibson. You'll find him at GRC.com, that's his website, the Gibson Research Corporation.

**Steve:** Well, and the good offices of Leo Laporte and TWiT & Company who all, all make it happen.

**Leo:** We provide the electricity.

**Steve:** And the bandwidth. And the personalities.

**Leo:** And the bandwidth. And half of the personality. One fifth of the personality. You can get, though - so at Steve's site he has kind of a little bit of - a little different flavor of Security Now!. He's got a 16Kb audio file, if you really have no bandwidth. And he also has a transcript. Actually, that's probably the lowest bandwidth version, written by an actual human being. Steve pays for it, so that's where...

**Steve:** Somebody asked for Morse code, and that would have been even lower bandwidth.

**Leo:** Mmm, yeah.

**Steve:** But I didn't think we wanted to do that.

**Leo:** Dit dit dot dot dash.

**Steve:** We could do an automatic - we could do an automatic translation, though, from...

**Leo:** I don't think Morse code is lower bandwidth. Isn't ASCII more efficient than Morse code? Get to work, Gibson.

**Steve:** I'll think about that.

**Leo:** So the longest character, what would it be? I don't know, four or five dots and dashes.

**Steve:** Oh, that's a good point because bits, yeah, eight bits is going to be a single char- well...

**Leo:** No, dots and dashes, if they're ones and zeroes, that would be more efficient.

**Steve:** Yeah, yeah.

**Leo:** Maybe half, half or less.

**Steve:** And also remember that we don't have compression with ASCII. All characters

are the same size. But Samuel Morse deliberately designed his code so that the most frequently occurring characters were the shortest.

**Leo:** Some characters are six. So the most would be six bits. The least would be three? I don't know.

**Steve:** Well, there's dit.

**Leo:** There's dit?

**Steve:** Yeah.

**Leo:** And da? Well, so there you go.

**Steve:** Yeah, "E" and "T" or some...

**Leo:** And there's no upper and lowercase.

**Steve:** Yeah. We were both - were you a Boy Scout? I was a Boy Scout. I had to - that was a...

**Leo:** You learned Morse code?

**Steve:** Dit dit dit dot da dit. Well, you - you're a ham. Didn't you have to do that?

**Leo:** You don't have to do it. That's why I became a ham. I would not become a ham when I had to learn Morse code. I stopped at Radio Telephone Operator 3rd Class. So, yes, Morse code would be the most efficient form. Take longer to listen to, though. Okay?

**Steve:** Oh, my lord.

**Leo:** I'm just saying.

**Steve:** [Vocalizes Morse code]

**Leo:** GRC.com. You can also go there to leave questions. We will do a Q&A episode, probably next week, in all likelihood. That's GRC.com/feedback. SpinRite's there, don't forget, the world's best hard drive recovery and maintenance utility. And lots of

other free stuff, too. The only thing that you pay for in that entire site is SpinRite. Everything else is free. GRC.com. He's on the Twitter, @SGgrc. Here at the TWiT website, TWiT.tv/sn, we have full-bandwidth audio and even video. Even hi-def video, if for some strange reason you want to see every pore. That's at TWiT.tv/sn. You can also subscribe, and I would recommend that. That way you miss not an episode, nary a Security Now!. Just go to your favorite podcatching app. iTunes is a good one. A lot of people use that. Stitcher is very popular all of a sudden. We're one of the top technology podcasts on Stitcher.

**Steve:** Yes. Security Then! is far less compelling than Security Now!.

**Leo:** It's got to be now. Get it while it's fresh, piping hot out of the editing ovens. Usually takes about two hours to get it out. Maybe sometimes a little longer on Tuesdays. That's when we do the show, Tuesday afternoon, 1:00 p.m. Pacific, 4:00 p.m. Eastern time, 2000 UTC. Thank you for joining us. Thank you, Steven.

**Steve:** My friend.

**Leo:** See you next time. And New Year's Eve.

**Steve:** Yes, indeed.

**Leo:** Bye-bye.

**Steve:** Bye.