Black Hat 2013, Tor & More

Description: With last week's Las Vegas Black Hat 2013 and DEFCON conferences just completed, Steve and Leo examine the most significant and worrisome revelations to emerge from that annual convocation, and also discuss and dissect the week's top security news.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-416.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-416-lq.mp3

SHOW TEASE: It's time for Security Now!. We were going to talk about PGP, but this week Steve found out so much stuff from Black Hat and DEFCON, he wants to talk about a big, big change in Firefox, and a flaw in Chrome that's not as bad as it sounds. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 416, recorded August 7th, 2013: Black Hat, Tor & More.

It's time for Security Now!, the show that covers you, your security, your privacy online with the Explainer in Chief himself, Mr. Steven "Honor Harrington" Gibson.

Steve Gibson: Actually, I think it's Steven "Peter F. Hamilton" Gibson, after...

Leo: Yeah. Did you finish?

Steve: Oh, I did. I finished at the end of last week, as I expected.

Leo: Steve was doing, what, 17-hour binges?

Steve: Well, no, it was a 17-mile walk on one day when I was reading with my Kindle.

Leo: That's right, yeah.

Steve: Basically they were 18-hour binges. I was just - I was reading, sleeping, and eating, and that's it. My friends were saying, what happened? Where did you go? I said, oh, I can't talk to you right now.

Leo: Oh, lord above. Oh, my, my. But you finished it.

Steve: And the problem is - I did. I did. The Void Trilogy, absolutely loved it. I can understand people saying, eh, it wasn't his best work. But if you just - if you're not in a hurry, if you don't, I mean, if you just want a really nice, rich, interesting storytelling and characterization in a sci-fi framework, I thought he did a really nice job.

Leo: What is your favorite Peter F. Hamilton? Do you have one?

Steve: Oh, I think it's got - well, the early Greg Mandel stuff is really fun.

Leo: Oh, good. I haven't read that. Ill have to get that.

Steve: And gland. "Gland" is the word I...

Leo: Gland?

Steve: "Gland" is the word I was trying to remember when - because Greg Mandel has an ESP gland...

Leo: Ah, yeah, that's right.

Steve: ...embedded in his head, and he squeezes, somehow he can...

Leo: Squeeze out a little ESP every once in a while.

Steve: ...squeeze out a little juice in his brain that...

Leo: Dribbles.

Steve: Yeah.

Leo: I'd like that.

Steve: Anyway, yeah, the esper gland.

Leo: The ESP gland would be nice.

Steve: Anyway, I think "Fallen Dragon," if I were to say...

Leo: I agree, I agree.

Steve: It's a reasonable size. It's beautiful characterization, really interesting story, fabulous ending. I mean, that's - if I were to complain about his really long works, like...

Leo: They just dribble off.

Steve: ...The Night's Dawn Trilogy was the first one, of course.

Leo: Don't like that one, really.

Steve: And then - and even the Void. It's like, he wrapped it all up, but it didn't have any, like, real kick and punch at the end. And, oh, my goodness, "Fallen Dragon" has a wonderful surprise twist that you never see coming. And so, yeah, I really just - I've read it several times. And I wish for amnesia so I could read it again.

Leo: I think my favorite trilogy is "Pandora's Star," the Pandora Star Trilogy.

Steve: Two books in that one.

Leo: Dilogy. Diology. Yeah. "Pandora's Star" and...

Steve: I agree. "Pandora's Star" as a substantial piece of work. Also his Commonwealth environment, his Commonwealth universe is really fun, the idea of having wormholes that you drive trains through because how could you better use a wormhole than by bringing back the train system. That makes total sense.

Leo: Yeah.

Steve: You know, so you have a transgalactic train system, courtesy of wormholes. It's just - it's great. So, and great - anyway. So the Commonwealth universe I really like that he develops with "Pandora's Star." And he is working on some additional ones. So, and I'm glad for that.

Leo: Oh, good.

Steve: But I'm not touching them for...

Leo: This is not the science fiction show, despite appearances.

Steve: No.

Leo: Now, last week you said we would do a PGP show, but current events have changed all that.

Steve: I should have known - well, yes. I should have known better. First of all, we've got the really interesting story of people discovering some spyware on their Firefox v17 browsers that are part of the Tor bundle which they are explicitly using for anonymizing themselves. It turns out our U.S. law enforcement has figured out how to get in there and break anonymity. So that would have been a big one.

Also Twitter just finalized yesterday, the second shoe dropped on their multifactor authentication. And there's something really interesting that they've done for in case you don't have your phone with you after requiring you to have your phone with you, which we need to talk about. And then of course Black Hat and DEFCON just happened. And I should have known better than to think that we were going to talk about anything else after that major convocation in Las Vegas.

Leo: Yes. I should have - I should have thought of that, too.

Steve: So this is Black Hat 2013...

Leo: This is our annual Black Hat show.

Steve: ...Tor & More podcast. And we will do a Q&A next week. And we'll start in on our - unless something else really significant happens. Anyway, but in general I want to talk about email security as our ongoing background theme until we have it completely wrapped up, with interruptions for user feedback and any other events that happen, the first one of which is this week.

Leo: Perfect. All right. Should we get to the Firefox story? Is that where you want to start?

Steve: Yeah. We just had, I think it was yesterday, Firefox v23, which was much anticipated. We've talked about it several times. And I'm sounding disappointed because I am. We once again failed to get third-party cookie blocking by default, despite the fact that they've punted, this is the second time, punted on that. There are several new features which are good. And we also already discussed the disappointing one, that an easy way to disable JavaScript was expected to be removed from the user interface, and it's gone.

Leo: Really.

Steve: So, yes.

Leo: That's odd.

Steve: And remember that what they did was, in updating, even if you had it turned off, and you update to v23, they turn JavaScript back on and then remove the ability to turn it off. Now, it's still, you can still get to it in the...

Leo: In about:config, now; right?

Steve: Exactly, in that ridiculous, I mean...

Leo: Chrome's doing that, too. It is so ridiculous.

Steve: It's an amazing number of settings that are available in there, which are sort of more like tweaks than regular UI things. Their feeling is this: JavaScript is now - it would be like turning HTML off, is what they feel. They're like, that setting is too potent. The web depends upon JavaScript now to the extent that you cannot really practically run without it. So they're saying we're going to remove it from the UI because it just no longer makes sense not to have JavaScript on all the time. And if you are security conscious because, oops, I mean, the way the FBI got into Tor was through JavaScript, and those - okay.

Leo: Whoa, you almost did a bad word.

Steve: I almost did a bad word.

Leo: Those what?

Steve: Monkeys. Monkeys.

Leo: The monkeys.

Steve: The monkeys. Even the Tor monkeys said, oh, you know, maybe we should have JavaScript on in the Tor browser by default. It didn't used to be on. And this would have never happened if they had left it off. But even they succumbed to, oh, well, let's - people are complaining.

Leo: Here's what I would say. You know, I think if you're sophisticated enough to be using Tor, then you're sophisticated enough to know what to do about Java. I can understand...

Steve: Well, no, but JavaScript.

Leo: I mean JavaScript, I should say. But I can understand why Firefox, which is a browser that they're trying to get everybody to use, might say, you know, if we let people just turn it off, they're going to - the experience is going to be so confusing that - but sophisticated users can either use NoScript, or you can actually bookmark the appropriate about:config statement to have a bookmark to turn it off and on if you want.

Steve: Well, and their point is exactly that. They're saying that - now, what's controversial is they didn't tell you that they were turning it back on.

Leo: Yeah. No, they should have said.

Steve: And that's - they really - that's...

Leo: You mean if I upgrade, it'll turn it back on if I have it off?

Steve: Yes.

Leo: Oh, that's not right.

Steve: Yes. Silently, with no indication. So their position is that, if you want JavaScript off, nobody can actually live with it off. So get NoScript. So they're, I mean, they officially say that's - we're removing the ability to turn it off. If people want control over JavaScript, you need to use a third-party plugin like NoScript, and that elevates you to an expert. And again, there are people in IT who are celebrating this because people had turned it off and then didn't know why those sites broke. I mean, first of all, it's not hard for a site to say, uh, hello.

Leo: Right, you need JavaScript, yeah.

Steve: You've got it turned off. This isn't going to work here with that turned off.

Leo: Well, and in fact there is a standard in HTML going back to 3.3 where...

Steve: Yeah, [indiscernible]...

Leo: ...you have a NoScript brack- you know, tag. And it says, hey, it says, I mean, it's so easy to implement. In fact, most of the time when you build scripting in, you always have, or I always do, a NoScript tag that says...

Steve: Deal with the case.

Leo: Yeah, you're not seeing this plugin because you have JavaScript turned off.

Steve: Yup.

Leo: Or offer, in a better world, offer a scriptless version of the same functionality, which would be also nice.

Steve: That, yeah, exactly. Now, and, see, I'm hardened because I, of course, I fly with scripting disabled. I'm using Firefox still because Google - unfortunately Chrome is just out of control in terms of bloat. I'm hearing a lot of people, by the way, saying, wow, Chrome is getting really slow because it's just getting so big. So at some point, I mean, and that happens. We saw it happen with Firefox. It's happened with Chrome. So at some point they'll need to just sort of - sort of do a reset. And they may be getting ready to do that as they move away from WebKit into their next rendering system. They may not be worrying about it right now because they're waiting for things to settle. But I love it from a security standpoint. Chrome's doing a lot of things. Although we're going to talk about a bit of controversy relative to passwords here in a second.

But Firefox - what I was thinking is I'm heartened by the fact that I am now frequently seeing sites that put up a little banner at the top of the page saying, oh, for full functionality of this site, you need to turn JavaScript on because it's not on right now. And I think, oh. And I look at the site, and I think, well, am I passing by? Do I care? Do I - is it worth - do I trust this site? And again, because of everything we've seen, it's not the site so much as everything about it because we've seen JavaScript code being injected onto otherwise good sites.

And in the news also this week we are seeing now malicious ad servers are injecting JavaScript into ads which are being served. So third-party ads being served up by good sites are running script now that can be malicious. So again, our audience knows how I feel about scripting. It's just absolutely it should be off unless you know you need to turn it on. And if you're skipping around the Internet and clicking on links, you really want to do that with JavaScript off because it's now the way bad stuff gets in. So that setting's removed from Firefox in v23.

There was an expectation that they were going to turn third-party cookies off by default, as Safari does. I initially tweeted yesterday the news that they had done that, along with another couple things. But then finally this morning I said, well, let me make sure about this because the wording is now misleading, the wording on their UI where they said "Keep cookies from sites you visit." And so I thought, oh, yay, that's exactly what Safari says.

Leo: And sites you don't visit.

Steve: Unfortunately, it also - and sites you don't, in the case of Firefox. Now, many people don't know, because this is one of those deep technology projects that I did where, after I solved all the problems, I got a lot of the documentation done, but then something came up and it took me off in a different direction. GRC has for years hosted a beautiful, if I do say so myself, cookie forensics system that instantly shows you exactly what your browser is doing. And because it's not linked to the main menu, nobody knows about it.

But I created a little bit.ly shortcut just now for my correction tweet when I realized that Firefox v23 had not blocked third-party cookies: bit.ly/checkcookie, all lowercase, c-h-e-c-k-c-o-o-k-i-e. And that just redirects you to my site's secure cookie checking system which needs to have browser redirects enabled because that's the technology it uses for causing your browser to query and get responses, query and get responses, several times as it tracks exactly what cookies are sent and received in order to do all kinds of cool things like show you whether you've got stale cookies, fresh cookies, no cookies. It actually plants cookies in icons, in iFrames, in page headers.

Leo: It does that fast. It's doing all that?

Steve: Yeah, oh, it does...

Leo: Geez, Louise.

Steve: It does a whole bunch of stuff and gives you a cool summary, which it then interprets all that for you in English, telling you what you need to worry about and not. And what's nice is for doing research in how various browsers are configured, it just allows you to quickly determine what's going on. For example, some browsers, even Safari, if you disable cookies and then do this, it'll say, well, new cookies are disabled, but old cookies are still being sent. And so you can go, ooh, and then close Safari and start it up again, and then see if that's still true. And then, like, wipe out cookies and then see. Anyway, it's fabulous for testing how cookies are handled. So bit.ly/checkcookie. And it's interesting because, when I tweeted this this morning, a bunch of people sent back surprises. They said, wow, I thought I had third-party cookies disabled, but they're not, and I didn't know that. So this little cookie forensics system that I built years ago when I was early on my third-party cookie campaign, it's always been there and works well.

So now they're saying that the Mozilla group are going to fall back to a new system involving cookie whitelisting and blacklisting, which is really disappointing. Stanford University has something called the Center for Internet and Society. And they're going to develop something called the CCH, the Cookie Clearing House, where they or someone decide which cookies you want and which ones you don't. And it's like, oh, oh, okay. So who knows. That's not ready yet. They've decided - apparently what was happening was, when third-party cookies were disabled, there were problems.

And we talked about this in our Q&A last week. We had a listener who wanted to tell IT how to fix the problem that they're having. And it is the case that you can, using some

settings, often enable or disable third-party cookies per site, very much like NoScript does with scripting per site, which would be a nice way to handle this. But it's a lot for the average user to deal with. So this has been discussed at great length over in GRC's newsgroups. And where many people settled out was to always accept third-party cookies, but only retain them until I close Firefox. That's sort of a nice compromise. Firefox does offer that, has offered it, still offers it. So that way third-party cookies are accepted during your use of the web browser. But when you close Firefox, it doesn't write them to permanent storage. They're never written to disk, and so they're discarded.

So that solves the long-term third-party tracking problem. The persistence of them is the problem. I mean, one of the other things that would be nice would be if you said we want third-party cookies to be transmuted into session cookies. Turn them into session cookies so they never live past the current session. That would work, too. But we don't seem to be given those sorts of options.

Leo: Well, at least when you restart the browser. That's actually surprisingly good. I didn't realize they were doing that. That's good.

Steve: Yes, and I like that. That solves the long-term persistence, which is really what the third-party cookie problem is, without messing up short-term gluing of third-party sites to your first-party sites.

Leo: Like the description the guy described last week that would work.

Steve: Yes, exactly.

Leo: Those iFrames would work, yeah.

Steve: Yes. And then, finally, they added - they've continued to march forward with mixed content. Remember that mixed content is where you're on a secure page, but not all of the URLs are secure. That's a problem because it means that some of the content coming back could have been messed with in addition to, of course, more easily spied upon, which is in the news currently, but could be altered. And you really don't want any of your content to be alterable. So it used to be that browsers would give you a popup and then would say - and most people just say okay, and then the mixed content page proceeds.

With v23, Firefox divides the insecure content which could be requested by the page you're visiting into active and inactive. It figures that inactive content, meaning static JPGs and GIF images and boilerplate body copy and things, if the browser is asking for those over an insecure page, Firefox is going to go, eh, okay, that's - it's not active content. But it's much tougher then on inactive, sorry, on active content. It blocks it. No notices, no warnings, it simply blocks it. It will not come up. And then a rather obscure new icon shows, or, yeah, well, a little image over in the very far left of your URL. It shows a shield that is half filled in. So it's like split, sort of a split shield. And that's their indication that we have blocked active mixed content on this page.

If you click that, it drops down a little information window explaining what the shield means and giving you the option of allowing all content, even active insecure content, on

this otherwise secure page. So that's what they feel now for v23 is the right way to handle the mixed content problem is to allow inactive, block active, but then sort of make a note of it over in the URL in case something seems wrong to you, and then you can click it and allow the active content, the active insecure component of the page, to come through. And a bunch of other updates and security fixes and so forth in v23.

So we're now at 23, and third-party cookies did not get blocked, and it doesn't look like they're going to be. I don't - it'll be interesting. Of course we'll be following the development of this CCH blacklist and whitelist. And I hope they do a good job. I mean, I guess anything's better than this right now, allowing them all by default. And having misleading text where they say, "Only from sites that you visit." It's like, okay, that really sounds like you're blocking third-party cookies. That's what Safari says, and that's what it means. But it's not what Firefox means.

Okay. Twitter has evolved their multifactor authentication. We covered the story that they had added SMS texting to add a factor to authentication. And they did that sort of in a hurry. In fact, there were people noting that there were employment ads for Twitter that were clearly looking for third-party authentication experts. So what they did is something completely different, which is a mixed blessing.

It would have been nice, in fact it would still be nice if they supported what everybody else has supported, which is OATH. Which is not OAuth, remember, it's OATH, O-A-T-H, which is the time-based one-time password. There are beautiful clients for our phones, so it's easy to have with you. We've talked about how nice it is. Everybody else, as they're coming out with one-time password solutions, is supporting OATH. But not Twitter. Twitter's rationale was that OATH inherently requires a shared secret. And we know that's true.

The idea is your phone has a gibberish-y looking, mumbo-jumbo, pseudorandom token, and that's the key that drives the sequence that OATH uses to generate, based on time of day, to generate the six-character password that changes every 30 seconds. The same gibberish is then maintained on the server side. So it knows what time it is. It generates, using the same shared secret, it generates the six characters that you should be giving to it, and often maybe plus or minus one set so that, if your clock is a little bit slow, and you happen to be sending it in exactly on the change point, it'll say, oh, yeah, well, close enough.

Twitter didn't want to have a shared secret because the danger is, if their server were compromised, and this is the whole thing, this is the whole rationale for not doing what everybody else is doing, if their server were compromised, those one-time passwords could get out. In which case a third party, potentially malicious, could generate the same token stream that you're generating. And so that would mean, if the compromise were not found - and in fairness to them, typically these compromises are often not found immediately. We often report that, oh, it was three months ago somebody got into the servers and got this stuff out, exfiltrated this data. And so the trouble would be that in that window between which the secret got out from the server and it was discovered as being loose in the wild, all hell would break loose.

So they're saying we don't want that responsibility. We want a system - and I can understand that. I can salute them. Also remember, if you were reusing that same token for multiple sites, then similarly to reusing your username and password, that they could, the bad guys who compromised, for example, Twitter's servers, wouldn't just have access to spoofing you on Twitter, but anywhere else you might have reused that same token, although the whole beauty of the OATH system and the clients that we like is that you've got a cool little lineup of tokens that are all changing at the same time. So they

said no, we're not going to do that. So they went with two things. One is very standard. The second is not, but very cool.

Okay, so the standard thing they did was a simple RSA key pair. So they're using public key encryption, whereas OATH is private key, symmetric key encryption. They added to the Twitter app for iOS and Android, and it's there now if you update your iOS and Android Twitter apps, they added yesterday a 2048-bit, so nice and strong, that's the strength we want now, and we'll be talking a little bit later about one of the presentations at Black Hat was disturbing because of recent advances we'll see in academic number factoring which begins to make people worry about how much longer RSA is going to be with us. So 2048-bit, nice long key pair. The private key never leaves your Twitter app. The public key is stored - I'm sorry. Yeah, the private key never leaves the Twitter app. The public key is what Twitter holds. However, all that allows them to do is authenticate your phone.

Now, that's good. Notice that that's the difference with OAuth. If you have the OAuth key, because you generate the same token the phone generates, you can both authenticate and impersonate. And so that's what Twitter wanted to avoid. If Twitter has the public key, they cannot impersonate. All they can do is authenticate. So when you want to log onto Twitter on a web server, and you've set this up in your phone, you give them your username and password to identify your account to Twitter. Then they see that this is set up, and they, Twitter, send your phone a "long random challenge," as it's called in crypto parlance, 190 bits, which is turned into 32 characters as a so-called "nonce," n-o-n-c-e, a pseudorandom nonce challenge.

Your phone, which uniquely holds a private key that was generated by the Twitter app, cryptographically signs the challenge, meaning that it probably hashes it and then encrypts it using the private key that it has, and returns it. Twitter, the only thing Twitter can do is verify the challenge by decrypting the signature that you made and verifying that it is the proper hash of the challenge it sent. Only someone having the private key can do that, so it says that very nicely validates you are you.

So that technology we have now. They like it because it avoids SMS and the possibility of SMS hacks. And as we said, for all the reasons I laid out first, it solves the problem of these public keys ever getting loose from them. All it can do is authenticate the challenge. But what if you don't have your phone? How do you do a recovery password? They recognize that all of these systems one way or another have to have a, yeah, but I left my phone somewhere, it's not with me, blah blah blah. So what they need is a backup code.

And this is where I have to say this is very cool. Back in 1996 - so what is that, 27 years ago? - Northwestern University published an idea, and this is back in, like in UNIX login days before the Internet, published a nifty concept that they called S/KEY, for Secure Key. And it was a one-time password solution that, for whatever reason, no one ever adopted. But it's nifty. So this system based on S/KEY is also in the new Twitter app for iOS and Android. And here is how it works: A separate - during this initial enrollment period, this enrollment and configuration with the new Twitter - I got myself distracted, sorry. The new Twitter iOS and Android app, when you're setting it up, a separate 64-bit random seed is hashed 10,000 times.

Leo: Wow.

Steve: Yeah, through SHA-256, so a state-of-the-art secure hash. It's hashed 10,000

times and turned into a 60-bit, 12-character string using something called Base32. And I think it was before we began recording, Leo, I mentioned that one of the first things I'm going to talk about when we start talking about email security is the whole...

Leo: ASCIIfication.

Steve: ASCIIfication, thanks, yes. Yes.

Leo: I just made that up. Turning stuff into ASCII.

Steve: And binaforcation [ph]...

Leo: Yeah, which is the opposite.

Steve: The process of going - you want reversible. You need to be able to turn binary into ASCII when it's necessary to transmit this over a channel that has limited - that isn't a binary-capable channel, and to turn it back into ASCII at the other end. We see that all the time in crypto with certificates because they're all binary. And we're going to be seeing it a lot with S/MIME and PGP and GPG and all those things. You see these blocks of what looks like gibberish in a piece of email. So I want to demystify what that is and get people comfortable with this whole process as we begin to plow into this. But this takes - so we do 10,000 hashes; reduce the result to a 60-bit string, binary; turn that into a 12-character ASCII string. That string is stored by Twitter on their servers. Okay?

Then the phone does the whole thing again, but stops one short. It does 9,999 hashes, does the same process, and gives you that 12-character string, saying write it down. This is your emergency recovery key. And so what's clever about this is you have the one preceding the one the server has. And we know hashing is one way. So if you ever need to log onto Twitter, and your phone is not available to you, but you wrote down this 12-character string and stuck it on a post-it note in your wallet, for example, then you go, oh, I don't have my phone with me, but I need to log on. So you type in the 12-character string and submit it.

Leo: Save it in your LastPass so it's secure.

Steve: That goes to - yes.

Leo: Right?

Steve: That goes to Twitter, that converts it back into binary, hashes it one more time to take it from 9,999 to 10,000, and it verifies that it matches the 10,000th string that you gave it. And then, when you ask your phone for another one, because you can only use it one time - oh, I forgot to say, after doing that, Twitter stores the 9,999th one because your next emergency code will be the 9,998th one.

So basically, as you use these over time, you're walking backwards through a chain of one-way functions, a chain of one-way hashes. As we know, it's always possible to go from the current one to the next one. It's impossible to go from - to predict what the predecessor was of the output of a hash. And so Twitter - so when you authenticate using your backup codes, you always give Twitter the one before, which it can verify is the one before the one it has, and then it stores that one as the one it has, and you'll give it the one before that next time.

Leo: Perfect.

Steve: Which is really cool, yeah.

Leo: I think, you know, it's something different, but it's a similar idea to what Apple and Microsoft do with the whole disk encryption built into the OS. They give you a way out if you lose the password. Right?

Steve: Right.

Leo: Yeah.

Steve: Right. And again, this also solves the problem of Twitter never having something that can compromise you.

Leo: Yes, that is clever.

Steve: All that can do is verify that you gave it the proper preceding passcode, and then it says, yes, you are you, logs you in, and then it keeps that one you gave it as the next succeeding passcode for the one that you're going to give it, which will precede it. So I thought that was okay. It's all new. They invented their own system. It would be nice...

Leo: And that's the shame, I mean, they didn't really need to reinvent the wheel.

Steve: Yeah, they wanted to own it themselves. And it would be nice, for example - oh, by the way, SMS is still supported. So if you don't want to do any of this, you can still use the existing SMS system. I think it would be nice if they also supported OATH that everybody else supports.

Leo: Yes, because I have my Google Authenticator.

Steve: Yes.

Leo: Works for LastPass, works for Google, works for Outlook.com. So, come on. If it's good enough for Microsoft and Google, do it. Seems...

Steve: Yeah. Give it to us as an option.

Leo: But, you know, Blizzard does that, too. They have - Blizzard's worse because they have their own authenticator.

Steve: Yup. They're like VeriSign, where you need to use their server in order to authenticate.

Leo: Yeah.

Steve: So, not surprisingly, we talked about this a week or two ago when DEFCON officially - was it DEFCON or Black Hat? Now I'm confusing myself because I wrote DEFCON...

Leo: Black Hat is the official formal conference, and then the hackers stick around for DEFCON.

Steve: Okay. So I'm...

Leo: And that's where they get crazy on it.

Steve: So what I wrote here was that NSA director Keith Alexander did not have a pleasant time.

Leo: That would be the Black Hat because he wasn't allowed to go to DEFCON. They uninvited the feds, yeah.

Steve: Correct. My notes confused me. Yes. So he was uninvited by DEFCON, saying, eh, you know - well, I mean, actually he wasn't, but all of the federal government was discouraged from going, with the argument that we need a little time to cool off here and so forth. So it was at Black Hat that he was, on July 31st, the opening keynote speaker. And so anyway, the crowd listening to his opening keynote was initially quiet and attentive and polite. But as he went on, basically trying to explain the how and why of the NSA's surveillance being both legal and effective, the crowd got increasingly restless, hostile, and heckling.

Leo: As am I, right now, just thinking about it.

Steve: Yeah. So that by the end it was, I mean, he did make it through his speech, but it was somewhat questionable whether he was going to be able to or not.

And then, on the Sunday show, I hope there's some follow-up to this because I would like to - and I ought to have done some more research. But it was just a comment in passing on ABC's "This Week" show with George Stephanopoulis where it came out that in 2011, so two years ago, the secret FISA Court produced a report themselves stating that what was being done was both unconstitutional and unlawful, and the report was classified and suppressed.

Leo: Oh, I just - this is endless. I don't know how it can get any worse.

Steve: So it's like, okay. So here's the FISA Court, being this court that operates already in secret, and then the judges themselves produce a report stating that what they're being asked to do, that is, that what they are overseeing and is being done is unconstitutional and unlawful, and that report...

Leo: Classified.

Steve: ...is suppressed, is classified and suppressed.

Leo: Holy cow.

Steve: That's really wrong, Leo.

Leo: We are rapidly approach 1984. I am very depressed. It's one thing if you can justify it and say, well, we're fighting terrorism. But doing something like that, where your own secret court condemns you, being suppressed, that's viciously illegal at this point.

Steve: And we have lawmakers, Congressmen, who cannot speak because the information they have is classified. And so they're unable to say this is wrong. Ugh, yeah.

Leo: Or they aren't being told, I mean, we're learning that, as well, that they are not being told everything...

Steve: In some cases they haven't been. But there are other...

Leo: There's vast illegalities going on, being perpetrated by the federal government, and not by the legislative branch, but by the executive branch and the law enforcement bureaus, vast illegalities, and no one is stopping it.

Steve: I tweeted a link, using bit.ly again, and I would be - I need to start using my own

system. I have the URL GRC.sc, for shortcut, so I could do my own little shortcuts. But nothing is going to interrupt SpinRite. I mean, I did, I did allow sci-fi to interrupt my work on SpinRite; but that stopped, and so I'm back. Anyway, so for now I'm using bit.ly. This is section215, all lowercase, is the tag: bit.ly/section215. This is the best, most succinct summary. Ars Technica just published this, which is a very good explanation for how this mass surveillance legislation was basically silently fudged.

And what you will see in there, Leo, you should read it sometime if you can, the lawmakers who watched this happen, who objected to it, who knew what was happening from the initial authorization, when it came up for reauthorization, some language was subtly changed that fundamentally changed what could be done. So, yeah. I mean, it's really, as you said, it's not good.

Now, in something that is sort of - I don't quite understand. This, for me, this is a tempest in a teapot. But Twitter doesn't think so, and the media doesn't think so. So 9to5Mac, their headline: "Security flaw in Chrome browser reveals plaintext passwords without authentication."

Leo: I was hoping you'd cover this because I'm very curious about this. And of course you need an expert to tell you.

Steve: And I thought, wow, okay, what is that?

Leo: Yeah. That's not good.

Steve: And so then they quote, they say: "The Guardian reports that a security flaw in Chrome allows anyone with access to a computer to view..."

Leo: Well, at least there are some requirements.

Steve: You've got to have a computer.

Leo: You could leave that whole clause out.

Steve: I know, "...to view all of the saved logins without requiring any form of authentication." And so then I go, what? So I switched over to the Guardian: "Google Chrome security flaw allows unrestricted password access." And then their subheader: "Plain text logon details for email, social networks, and company systems stored in the browser's settings panel." And I thought, well, yeah, okay. So the Guardian - and this sort of, this explains both sides of this, and some sort of, like, this sets it up, so I'm just going to share this Guardian story.

"A serious flaw in the security of Google's Chrome browser lets anyone with access to a user's computer see all the passwords stored for email, social media and other sites, directly from the settings panel."

Leo: What?

Steve: "No password is needed to view them." Okay. "Besides personal accounts, sensitive company login details would be compromised if someone who used Chrome left their computer unattended with the screen active. Seeing the passwords is achieved simply by clicking on the Settings icon" - and Leo, you can follow along.

Leo: I'm doing it right now, yes, terrified.

Steve: Oh, yeah. Are you sitting down? Are you on your ball? Are you centered on your ball?

Leo: Okay. I am centered because this is horrible, yes, yes.

Steve: Good, yeah, because if you were off-center on your ball, you might lose it.

Leo: Yes, okay.

Steve: Okay, "...choosing 'Show advanced settings.'"

Leo: Show advanced, oh, that's way at the bottom here. Okay, good.

Steve: Oh, yeah. Go all the way, Leo, it's a long scroll to the bottom.

Leo: Passwords and forms?

Steve: Yes, yes, that's where you want to go.

Leo: Yeah, yeah, yeah.

Steve: And "Manage saved passwords..."

Leo: Okay, let's click that.

Steve: "...in the 'Passwords and forms' section."

Leo: I'm going to take the camera off of it because I don't want to reveal anything.

Steve: No. "A list of obscured passwords is then revealed for sites."

Leo: Yeah, they're obscured.

Steve: They're little black dots, Leo.

Leo: Yeah, little black dots, yeah.

Steve: And they're all the same length, aren't they.

Leo: No, they're different lengths.

Steve: Oh, oh, there's some information leakage right there.

Leo: Yeah, that's not good, yeah.

Steve: No. "But clicking beside them..."

Leo: Okay. I'm going to hide this again. Clicking beside it...

Steve: "...reveals the plain text..."

Leo: Oh, the Show button, yeah.

Steve: Leo, there's a Show button. Oh, my god.

Leo: And it does. It shows it.

Steve: Who put that there? Oh, my, that's a serious security flaw.

Leo: Yes, yes.

Steve: Yes. And now there for anyone to see is your password.

Leo: Yes.

Steve: That could be copied, Leo.

Leo: It could be.

Steve: It could be sent via a screenshot.

Leo: Yes.

Steve: To an outside site.

Leo: It sure could.

Steve: My god, it means pandemonium.

Leo: But you have to be logged in as me, on my version of Chrome.

Steve: And, and, and, and...

Leo: And have access to my computer.

Steve: But get this, Leo. But the head of Google's Chrome development team, Justin...

Leo: If I can't look at my passwords, that could be a problem, folks. I want to be able to look at my passwords.

Steve: And get this. This is where it just goes.

Leo: I could do that with LastPass, too.

Steve: The head of Google - yeah, I know - Google's Chrome developer team said he was aware of this.

Leo: Yeah, I'm aware of it. What of it?

Steve: Uh-huh, and there are no plans, Leo...

Leo: No, no, no plans.

Steve: ...to change the system.

Leo: No, no plans at all.

Steve: Now, however, now, the Guardian took a little - in the middle of generating a great story here with a fabulous headline, he goes on to say: "That response was described by Sir Tim Berners-Lee..."

Leo: The guy, yeah, the man.

Steve: "...the British inventor of the whole web..."

Leo: The whole thing, he invented it.

Steve: "...as 'disappointing.'"

Leo: Come on, Tim.

Steve: Sir Tim is disappointed.

Leo: Tim obviously didn't really look into it.

Steve: "He characterized the flaw" - Leo, you have just demonstrated a flaw of Chrome security - "as 'how to get all your big sister's passwords,'" says Tim.

Leo: Yes, that's right. If your big sister's stupid enough not to log off.

Steve: Just to give you some background here: "Chrome is one of the three most widely used browsers." I'm still quoting the Guardian.

Leo: No, really?

Steve: Yes, from the Guardian. The Guardian says: "...the three most widely used browsers on desktops worldwide, along with Microsoft's Internet Explorer and Mozilla's Firefox. It has millions, Leo, of users and is seen" - the Guardian didn't say "Leo," I added that - "is seen by some as crucial to Google's future efforts to monetize web use."

Leo: It's crucial. I mean, I'm just curious, because if I go to LastPass in my browser, and then I look at a password - here's let's pick something - and I say let's edit that

password...

Steve: I did that yesterday, Leo.

Leo: And then - ooh, let's not show that. Oh, that was a mistake. I just showed my credit card number. I'm going to have to change that. This is what happens when we start doing this stuff. It had my credit card number in the clear. How dare they?

Steve: After you asked it to show it to you.

Leo: How dare they? Yes. Yeah, it had the whole thing. I guess I'm going to have to change that. Oh, well.

Steve: So, folks, there you have it. If you want to know what your passwords are, Google will show them to you.

Leo: Did you get the screenshot, those of you at home? Okay.

Steve: Now, what Firefox does is offers you the option which Google has left out, and this seems to be the focus of this whole tempest in a teapot. Firefox will allow you to create a master password to protect all of your mini passwords, your non-master, your slave passwords. Chrome doesn't do that. Everybody would be happy, apparently, if Chrome allowed you the option of creating a master password. But this Justin, Chrome's head of - he's the head of Chrome's developer team, wrote on Hacker News that, quote: "We've also been repeatedly asked why we don't just support a master password or something similar, even if we don't believe it works. We've debated it over and over again" - so, folks, this was not left out by oversight - "but the conclusion we've always come to is that we don't want to provide users with a false sense of security and encourage risky behavior. We want to be very clear that, when you grant someone access to your OS user account, they can get everything."

And what I loved was this article finishes with an unnamed security manager at a publishing company, who said: "The fact you can view the passwords means they are stored in reversible form, which means that the dark coders out there will be writing a Trojan to steal that password store as we speak." Now, this is not a very smart security manager, unnamed, thankfully, at a publishing company, because of course they're reversible. They're stored in your browser so that your browser can send them to the website, as if you had typed them.

Leo: If it didn't do that, you'd - okay.

Steve: Okay.

Leo: So, by the way, don't use that credit card I just put out over the air because I've replaced it, so - I can't believe I did that. So, now, the same thing happens with LastPass. But you have to be logged in in both cases. You have to be logged in to Chrome, too.

Steve: Browsers want to - browsers have always offered you, would you like me to save the password of this page?

Leo: Right. And Firefox will password protect that. Actually, Firefox did used to store that without - in a file without encryption. But that was a while ago.

Steve: Yes. This has been resolved quite a while ago. But mostly, I mean, the lesson we - our listeners know, yes, you are a privileged user of your computer, which is why I hope you have a logon password for your computer. What that means is you should shut it down or lock the system if you're going to be away. Someone sitting at your computer is presumed to be you, at a level of granularity you can control. Under LastPass you can say "Prompt me for my master password" every time LastPass is going to decrypt your local store in order to send a password off to a website. Or you can say, no, I'm the only one around. Ask me only when I fire up Firefox. Or make that, even that, make that more persistent. So you can control that level. And so what Google's position is, our feeling is none of that is really very useful. We don't want to pretend that it is. So, yes, we'll show you your password if you ask us. And of course...

Leo: Yeah. If you have physical access to a computer in general...

Steve: Yeah, you're god of that machine.

Leo: Yeah. I mean, come on. That's just moronic.

Steve: Yeah. Yeah.

Leo: Yeah. And speaking of moronic, that was really stupid, to show my - all that information. God.

Steve: So one of the other Black Hat presentations was a disturbing analysis of the current academic research in factoring. And essentially a group of guys at a firm, iSEC Partners, did a very chilling presentation showing the advances just in the last six months on edge cases of factoring. They don't fully apply today, so it's not like a linear scale where we're linearly getting better. But the mathematicians are really intrigued by the idea. And when you get a whole bunch of mathematicians really intrigued, and you allow them to talk to each other and publish papers and go, oh, look what happened, we realized that this little subdomain over here has a recurring modulo in the field of something or other, then other researchers go, hey, hmm, that kind of fits in with what I was doing, and they end up whittling away at this. This is the way it's going to happen.

So right now the Diffie-Hellman key exchange, which relies on the so-called "discrete logarithm problem," which we've discussed, that was released in '76. And RSA encryption, which relies on the hardness of factoring problem, that happened in '77. So that's more than 30 years ago. So 30 years ago, I mean, 30 years from then was 2007. And that's around the time that Elliptic Curve Crypto, ECC, began to happen. So, really, 30 years is a good run. That's, I mean, that's a successful lifetime for anything in the security arena.

And these guys argue that the problem is we are not prepared to move as quickly as we should. RSA and Diffie-Hellman are too well entrenched in the crypto ecosystem such that, if factoring fell tomorrow - and they're not saying it's going to, but Bruce Schneier's famous quote was "Attacks never get worse, they only get better." And one of our big topics for today is BREACH, a new attack on SSL encryption, HTTPS sessions, where we're going to see exactly that. So they ended their presentation showing timelines and exponential graphs and charts, and they really made a strong case for the fact that we need to move.

Now, there is something interesting holding us back, and that is that BlackBerry, of all people, own the patents on ECC. And the NSA has even licensed the patents for some domains of use from BlackBerry. But ECC is somewhat patent encumbered. And so in general the industry doesn't like freely using and relying on patent-encumbered crypto. I mean, that's always been a concern, the issue of intellectual property. So they made a plea for BlackBerry to consider doing what's right and allowing some use of their patents. And one wishes that BlackBerry were in better shape right now because they're probably looking at their elliptic curve crypto patents covetously and thinking, oh, how much money can we get for these? Which is not good.

But so just it was an interesting presentation, saying, you know, keep an eye on RSA. The problem is making the keys bigger, the RSA keys. We're talking now about 2048 bits. Google famously is retiring its 1024-bit keys, moving to 2048. Our own backend credit card processing merchant gateway recently sent out email saying that their test gateway is now running 2048. By a couple months from now, they're going to be switching over. So anyone relying on, anyone who might have SSL crypto which is sensitive, and my implementation is not, but so there's another instance of everyone beginning to have to move to 2048 as their 1024-bit certificates are expiring.

So the problem is that elliptic curve crypto is much faster and uses much shorter keys. You get about the equivalent, with only twice the key length, of elliptic curve crypto, which is a public key - it's an asymmetric key technology. You only need to go, to get about the equivalent security of a 256-bit key in symmetric, you only need about a 512-bit key in elliptic curve; whereas you need about a 16,000-bit key using prime factorization RSA. And, boy, things really slow down. That's the problem is RSA does not scale well in key length and performance in order to maintain security, whereas elliptic curve crypto scales much better, much more efficiently.

I did also note, just in passing, that even Wikimedia is responding to the NSA spying news, and they are hastening their move to HTTPS Everywhere. They're going to be switching to a full secure mode just so that - to make it less easy to have the users of Wikipedia and other Wikimedia properties surveilled.

And then, lastly, before we get into our other technology stuff, I noted, and many people brought it to my attention, that the other agencies, almost predictably, the other government agencies are now complaining that the NSA has access to information they would like to have because it would make their...

Leo: Well, of course.

Steve: ...discoveries and cases. I know. The New York Times said: "The National Security Agency's dominant role as the nation's spy warehouse has spurred frequent tensions and turf fights with other federal intelligence agencies that want to use its surveillance tools for their own investigations, officials say. Agencies working to curb drug trafficking, cyberattacks, money laundering, counterfeiting, and even copyright infringement..." - we know who they are.

Leo: Yeah.

Steve: Uh-huh, "...complain that their attempts to exploit the security agency's vast resources have often been turned down because their own investigations are not considered a high enough priority, current and former government officials have said. Intelligence officials say they have been careful to limit the use of the security agency's troves of data and eavesdropping spyware for fear that they could be misused in ways that violate Americans' privacy rights." Uh-huh.

Leo: What a surprise, yeah.

Steve: We've got it, and you guys don't get it.

Leo: Except that I don't buy it.

Steve: No.

Leo: I'm sure everybody gets it.

Steve: So two brief notes. I did, as I mentioned at the top of the show, finish The Void Trilogy last week, loved it. And I would recommend "Fallen Dragon" as anyone's first dip into Peter Hamilton, followed by the "Pandora's Star" and "Judas Unchained" pair of books. "Elysium," Matt Damon stars, and opens on Friday.

Leo: I can't wait. That might actually be good.

Steve: Looks great, from what I've seen. And "Oblivion," the only movie that I've seen twice this summer, was released on disk.

Leo: I thought you didn't like it.

Steve: Oh, I loved it.

Leo: Oh, you liked it. That's the Tom Cruise one.

Steve: Yeah, thought it was great.

Leo: Oh, okay.

Steve: I really enjoyed it. And then just totally random, but I got a chuckle out of this, I got a note from Amazon telling me that the extended version of "The Hobbit" was just out on DVD.

Leo: In case it wasn't long enough.

Steve: And I'm thinking, what? They extended the extended movie? Because, I mean, everybody was complaining.

Leo: It was already three hours or something.

Steve: Because the movie was ridiculously long. But, no, we didn't see it all, Leo. There was some that was left on the cutting room floor. So they've pieced it back together, and the extended version...

Leo: Waste no furry toe. So that's annoying because of course I bought the deluxe Blu-ray of "The Hobbit" when it came out, and now - that's probably why. They want me to buy it again.

Steve: And did you watch it yet?

Leo: Yeah, several times. I liked it.

Steve: Yeah. I enjoyed the movie also.

Leo: I didn't dislike it. I mean, it was nothing like the book, but I didn't dislike it.

Steve: I didn't think it was too long.

Leo: New one comes out in December.

Steve: You can get more if you want. It's extended now.

Leo: I can only imagine.

Steve: And I am back working on SpinRite.

Leo: Yay.

Steve: I'm working now on the low-level driver for the hardware, which will give us the access to huge buffers and was going to just really make SpinRite, going to be a huge performance benefit for SpinRite. What I did so far, there's a bunch of very delicate timing in the so-called ATA specification, where, for example, if you change the drive you're selecting, where you've got two drives on a traditional flat cable, the IDE-style drive with master and slave, you have to wait 400 nanoseconds for the newly selected drive to put its status on the bus, and the drive that was selected to release its status from the bus.

Leo: 400 billionths of a second?

Steve: Yeah.

Leo: Oh, dear.

Steve: Yeah. And the problem is it's not easy to know when that amount of time has passed.

Leo: I don't think you can count it, really.

Steve: Well, and so it's, like, easy to know if much more time has passed. But if I did that, then I wouldn't be running as fast as I could. And my whole, as is always the case for me, I want to write this code once and have it last forever. So I've seen - and when I've looked around at how other people have solved the problem, I have been disappointed because even on really good websites they say, "Issue five output instructions and only use the" - or "five input instructions sampling the status, and only use the results from the last one." And it's like, what? Well, how do you know how long each output instruction's going to be? Because that keeps changing and getting faster. And it varies whether you're on an old-style motherboard or a fancy fast chipset and so forth. So I've been very disappointed with what I've seen.

Anyway, what I wrote and nailed down after I got back to SpinRite was a system that gives me accuracy on the test platform I have, and it will generally be the case, down to 323 picoseconds of timing so that the 500-nanosecond, I'm going to give it an extra hundred just because that's wise, the 500-nanosecond delay requires 1,549 of those 323-picosecond intervals. So I've got that nailed down. I have a general purpose, highly accurate, high-resolution time base which establishes itself in 3.5 seconds on anyone's machine. And then that will be driving all of the access where software's involved and waiting for things to happen on the hardware bus. So it's going to be good. That's going

to work.

Leo: How fun.

Steve: Oh, yeah.

Leo: It's fun to be solving problems like this. That's why you do it.

Steve: Yup. It is exactly why I do it, because I want to - it's stuff I haven't done before, and then I come up with exactly the right solution that nobody else seems to have taken the time to do right.

Leo: Gibson [indiscernible].

Steve: So everybody who has SpinRite 6 gets the new one.

Leo: Yay. Leo Laporte, Steve Gibson, and there's still lots more to talk about.

Steve: Actually two major issues, which I think will fit nicely into this final 20 minutes. So responding to an arrest warrant and U.S. extradition request, an arguably slimy guy, Eric Eoin Marques, M-a-r-q-u-e-s, was arrested in Ireland last Thursday. And a paper over in Ireland, the Irish Independent, reported that Marques is wanted for distributing child pornography in a federal case filed in Maryland, in the U.S., and quotes an FBI special agent describing Marques as, quote, "the largest facilitator of child pornography on the planet."

And elsewhere, I don't remember now where - oh, here it is, in an article by Wired. I was going to follow into this. Wired magazine then picked up the news, which was just about coincident with this event, and so as a consequence it's not believed to be coincidental: Freedom Hosting, which was the facility that Eric was operating, began having, well, it put up a notice saying that the sites it was hosting were temporarily down for maintenance, and spyware was being injected.

Freedom Hosting is in the Tor network using the Tor hidden services that we have talked about. In fact, we did a podcast on using the distributed hash tables and how the use of hidden services was sort of an inversion of what Tor was originally designed to anonymize. Tor initially was anonymizing users who would use the Tor client to access the Internet through multiple Tor nodes, hopping around with the onion layers being peeled each time, decrypting their traffic, so that no - where no node needed to be trusted. Very clever system. Then they added a second facility such that, rather than having your traffic eventually go out onto the Internet, where then it would be public, it was possible to similarly conceal servers such that they would have name.onion, so they used the .onion top-level domain, and the server itself could be hidden within the Tor network.

Now, it's very valuable, of course, for many good purposes. And unfortunately, it can also, as all of these technologies can, just like cryptography itself, can be used for bad

purposes. And there were - apparently Freedom Hosting was hosting websites that were offering a great deal of child pornography.

So Wired's reporting said: "Freedom Hosting has long been notorious for allowing child porn to live on its servers. In 2011, the hactivist collective Anonymous singled out Freedom Hosting for denial-of-service attacks after allegedly finding the firm hosted 95% of the child pornography hidden services on the Tor network. Freedom Hosting is a provider of turnkey" - this is Wired, continuing - "Freedom Hosting is a provider of turnkey 'Tor hidden service' sites special sites with addresses ending in .onion that hide their geographic location behind layers of routing and can be reached only over the Tor anonymity network. Tor hidden services are ideal for websites that need to evade surveillance or protect users' privacy to an extraordinary degree - which can include human rights groups and journalists. But it also naturally appeals to serious criminal elements.

"Shortly after" - this is still Wired. "Shortly after Marques's arrest last week, all of the hidden service sites hosted by Freedom Hosting began displaying a 'Down for Maintenance' message. That included websites that had nothing to do with child pornography, such as the secure email provider Tor Mail." And that's been one of the aspects of controversy here is that, as has happened before, this was a bit of a blunt instrument that it turns out U.S. law enforcement was using. We'll get to the technology of that in a second.

"Some visitors," says Wired, "looking at the source code of the maintenance page, realized that it included a hidden iFrame tag" - and here we go back to what I was saying last week about iFrames just being fundamentally a bad idea, or dangerous - "that loaded a mysterious clump of JavaScript code from a Verizon Business Internet address located in Virginia."

So first off, remember that what iFrames do is they're like a web page within a web page. They are a frame that has a URL, and they will induce the browser to go load that page with whatever the page contains, in this case JavaScript from a Verizon Business Internet address. Now, there's many layers to this, and I alluded to one earlier. One problem was that the Tor Browser Bundle, which is the TBB, the Tor browser bundle is what most people use for accessing Tor, including these hidden servers, had switched its policy sometime in the past from disabling JavaScript to enabling it, specifically because sites weren't working. So the first part of this vulnerability was JavaScript was enabled. And there was a known vulnerability, since patched, which was not propagated into the Tor browser bundle because this was a down version, v17 of Firefox, and this was the extended service release, the ESR version, which did not do automatic updates. So it wasn't getting updated.

So, "By midday on Sunday," so this is like three days later, "the code was being circulated and dissected all over the 'Net. Mozilla confirmed the code exploits a critical memory management vulnerability in Firefox that was publicly reported back in June, on the 25th, and is fixed in the latest version of the browser."

The Tor Project wrote in a blog post Sunday: "The malware payload could be trying to exploit potential bugs in Firefox 17 ESR, on which our Tor browser is based. We're investigating these bugs and will fix them if we can."

And so Wired concludes, saying: "The inevitable conclusion is that the malware is designed specifically to attack the Tor browser." I agree. "The strongest clue that the culprit is the FBI, beyond the circumstantial timing of Eric Marques's arrest, is that the malware does nothing but identify the target." And so some people have reverse-

engineered the code. I have looked at it. It is Windows shellcode, essentially. The JavaScript has a variable called "Magneto" which contains the Windows executable shellcode. It uses a memory management flaw in JavaScript to inject this Windows shellcode into memory and then execute it.

What it does is it connects to an IP address, 65.222.202.54 over port 80, which is to say a web port. And then it assembles and issues a simple HTTP query to that IP address providing the system's hostname and the local network adapter's MAC address. And what's clever about that is, as we have often discussed and have explicitly talked about when we've been talking about the Ethernet protocol, which is where the MAC address lives, unless they are manually changed, they are globally unique. So what is collected by a server, an HTTP server at that IP, is the user's IP address, because that's where the non-spoofable TCP connection comes from, the system's hostname, and the probably globally unique, or if nothing else, the current MAC address of that machine.

So I've seen all kinds of misreporting about this. There was some - some reports believed that this was part of the CIPAV, which is a well-known, for about a decade now, FBI spyware tool. CIPAV stands for Computer and Internet Protocol Address Verifier. The reason I think that's misreported is that that is a much more comprehensive piece of spyware. It collects the IP address, the MAC address, the list of open TCP and UDP ports, the list of running programs, the OS type, version number and serial number, the default Internet browser and version, the current user of the system, the current logged-in username, and the last visited URL. So it's a much bigger blob. This exploit was much more tightly written and much smaller, and it doesn't leave any kind of backdoor or any other modification behind. And actually it cleans itself up and hangs the system so that, essentially when it's restarted...

Leo: This is a reboot.

Steve: ...it forces a reboot. It cleans itself out. So it seems very minimal. And back, oh, I can't think of now when it was, but the Ninth Circuit Court of Appeals did rule that this minimal level of identification did not meet the level of privacy that a user on the Internet could reasonably expect to have, and it was constitutional to collect this without first having a search warrant.

Leo: I think so. I mean, your IP address is exposed.

Steve: Yeah, and your MAC address and...

Leo: And your MAC address. It's just that you've been trying to prevent that using Tor.

Steve: That, and so - exactly.

Leo: And this is through the Tor.

Steve: Exactly. And so that's the key here is, remember that you're using Tor, and this

client, and there's a tremendous amount of technology involved in hiding you. You've got all these wrappers of onion layers, and your data is bouncing around among Tor browsers, and onions are being pooled and so forth. And then your computer does a straight hot beeline connection to the FBI. And in fact there were early reports that this was an NSA IP. They didn't turn out to be true. There were reports that it was an SIAC, which is a major government contractor. That's not the case. Nobody knows where this goes. It turns out that it's been the IP address, 65.222.202.54, has been traced to a small block of eight so-called "ghost IPs." It's just a, quote, "unallocated block" at the Verizon Business Datacenter in Ashburn, Virginia. So...

Leo: So nobody's using it.

Steve: Yeah.

Leo: Nobody's using that.

Steve: There is a server. There's a server that accepts, happily accepts your TCP connection, Leo. And so essentially what this - the whole takeaway is that this bad guy, arguably bad guy, providing a huge amount, presumably for money, I mean, he's charging, I would imagine, I mean, this is a profit basis for him, selling kiddie porn, is extradited and arrested. The FBI figures out where his machines are, immediately puts up a "sites are down" message, but inserts spyware such that anybody receiving those pages, anybody seeing those pages will essentially completely de-anonymize themselves, connecting to the server in Virginia with their IP, their MAC address, their machine's name, and now they're in trouble. So as well they should be, although people using Tor Mail, again, this did cut a large swath. Basically everyone using this Freedom Hosting who visited - oh, by the way, had to be on Windows. If you were on a Mac, the shellcode would just explode and do nothing.

Leo: Clever.

Steve: So, yeah, clever.

Leo: Yeah. Can't really complain about it, since it was...

Steve: No. I don't - I would say this was - if nothing else, our takeaway here is that the FBI's got some smart people, too, and they are on the ball, and they are using this technology, and they're going to where law allows them to. And this did not cross a boundary. This was only getting people who were going to these sites, de-anonymizing them, and then allowing the FBI to then get a search warrant to get the person's machine and verify that in fact they were doing this, they had this behavior in their recent history.

Leo: Very interesting.

Steve: And lastly, wrapping this up, we discussed at some length in the past Thai Duong and Juliano Rizzo, one of whom was at the beach looking at bikinis, everyone will remember...

Leo: Oh, yes [laughing].

Steve: ...when they came up with their so-called CRIME, C-R-I-M-E is the acronym, for their very clever approach for using compression, essentially using compression variation, the variation in the amount that plaintext would be compressed to determine what browser query headers were. What we know of compression is the more redundancy you have, the greater the compression. So the theory was, and what they did was they created an absolutely robust attack. The theory was we should be able to inject different things and then look at how the SSL/TLS compression compresses our stuff plus the stuff we want to know. If our stuff matches the stuff we want to know, the compressor will compress it well because there's not much entropy there. If our stuff doesn't match what we want to know, it will expand it and not compress it.

So they turned this into the CRIME attack, where shortly afterwards everyone stopped using SSL/TLS compression. Now, that was easy to do. No one uses it anymore because of these guys back last year. But nobody was really using it the first place. It was in the spec. Some people had it on. Newer browsers used it or offered it. Newer servers offered it. You had to have it. It was one of those things where you had to have it at both ends so that both ends would negotiate. And if they both supported it, then they would turn on. But it was no big deal. Why? Because most of the traffic is going the other direction. You send a URL to the website, and wham, back comes the page. The page is where all the data is. And CRIME was only an attack on the query headers. What they got was, and they demonstrated this, they got the session cookie. So that was certainly valuable. But that's where their attack ended.

BREACH, B-R-E-A-C-H, Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext - nice acronym, BREACH - was revealed last week at Black Hat. BreachAttack.com has all the details, if you want more than you get here. And basically these guys figured out how to go the other direction. And the bad news is everybody uses compression coming from the server. What CRIME did - remember that we're all sort of talking about layers. In communications technology we've got, like, Ethernet at the bottom, and below that is, like, ARP, for example, for the Ethernet packets to route. And then we've got the IP layer, then we've got the TCP layer, then we have the SSL layer, and then we have on top of that HTTPS, essentially. So the SSL/TLS layer really didn't matter because all it was was query headers, essentially, going in that direction.

But these guys have figured out how to do the same thing in a relatively short time to crack critical data, state data, being sent in the response headers, which often contains cookie setting material, or it can offer redundantly contained session-based material. That stuff is going in the other direction. Everybody compresses their responses. The reason is, as we've talked about this before, HTML is incredibly redundant. It's a markup language, hypertext markup language, HTML, which has so much, not only is it often in whatever language the text is written in, and language is inherently redundant, but all of this markup stuff, you know, tags are used with great abandon. And they can all be compressed.

Consequently, you get, for example, you can take a typical HTML page and squeeze it down to 20 to 25% of its original size. So that's huge in terms of performance, in terms

of minimizing delivery bandwidth, and also improving the whole response time system of the web ecosystem because that means that the page gets to you in one quarter to one fifth the time. That contains links that allow the browser to ask for all the other assets of the page much faster and get them all coming to you. So the idea that we can no longer use HTTP compression is horrifying.

And what these guys demonstrated was an attack on - I can't remember now whether it was IE or Chrome. Both were cited as being readily vulnerable because they're very fast. And in the summary of their talk it said: "In this hands-on talk, we will introduce new targeted techniques and research that allow an attacker to reliably retrieve encrypted secrets - session identifiers, tokens, OAuth tokens, email addresses, view state hidden fields, et cetera - from an HTTPS channel. We will demonstrate this new browser vector is real and practical by executing a Proof of Concept (PoC) against a major enterprise product in under 30 seconds." The title of their talk was "SSL, Gone in 30 Seconds, a BREACH Beyond CRIME."

Leo: That's better than stealing a car.

Steve: And they said: "We will describe the algorithm behind the attack, how the usage of basic statistical analysis can be applied to extract data from dynamic pages, as well as practical mitigations you can implement today. We will also describe the posture of different Software as a Service (SaaS) vendors vis--vis this attack. Finally, to provide the community with ability to build on our research, determine levels of exposure, and deploy appropriate protection, we will release the BREACH tool."

Leo: Oh.

Steve: Yes.

Leo: Wow.

Steve: Yes. So that has happened. For mitigation, they don't offer much. No. 1, disable HTTP compression. Ouch. Nobody can afford to do that today. Disconnect secrets from user input. Somehow keep them, like minimize the exposure. They suggest masking the secrets. For example, randomizing them with XOR and something else which you provide so that the idea is you don't want something static to be sent from the server over and over and over. So, for example, don't redundantly provide the session token over and over and over and over because that's the kind of thing that's vulnerable. And so essentially we've all - all web browser developers have assumed that, if they enforce HTTPS, that is, SSL sessions, on the delivery of a secure page, then nothing else matters. They can just - they can send cookies. They can have headers that they would not ever want anyone else to see. They don't have to worry. That's gone now.

Now, any webmasters should immediately take a look, who are concerned about this, take a look at what they're sending out in the response headers, and do what they need to to obscure them - rearrange them, pad them, add pseudorandom noise as the first header, append random varying length junk before the actual secret content. It's really, from this point on, it's going to be necessary to deliberately obscure the response headers with stuff that will defeat this kind of a multiple query - this is using

compression, the same concept as CRIME, where that's why turning off compression saves you is these guys are injecting stuff and seeing how the compression changes based on what they inject. That allows them to essentially infer what was already there being compressed.

And maybe we could have an evolution of compression so that it's not as deterministic. Maybe there could be some change to HTTP compression to thwart this kind of attack. Absent that, and until then, it's going to be necessary to do something to obscure these headers. So a word of warning to all webmasters out there. And there's our podcast.

Leo: And a lovely one it was indeed. Well, you've given me great reason to be depressed. But thank you anyway.

Steve: Yup. Lots of news from Black Hat and lots of security stuff.

Leo: Wow. Steve Gibson does this show every Wednesday, 11:00 a.m. Pacific, 2:00 p.m. Eastern time, 19:00 UTC. That's at TWiT.tv. I hope you'll join us live. If you can't, well, you can always get audio or video on demand. There's several ways to do that. GRC.com has 16Kb audio and the smallest form of all, the text transcriptions by Elaine. GRC.com, that's Steve's site. Or you have, at TWiT.tv, high-quality audio and video, or wherever finer podcasts are available. Do you make a feed available of the 16Kb?

Steve: No, don't do a feed.

Leo: No. So you just have to download that directly.

Steve: Yeah.

Leo: Good. You won't get sued. And that's - forget I said that. We'll talk another time. What else is there? Well, of course, last - not LastPass, SpinRite. Another two-syllable, must-have utility.

Steve: What else is there?

Leo: What else is there?

Steve: Ah, yeah.

Leo: SpinRite, the world's finest hard drive maintenance and recovery utility. And a lot of freebies, too. So go over there and visit: GRC.com. Will you do questions next week, you think?

Steve: Yeah, let's do Q&A next week, and I'll probably have some more really interesting SpinRite news, too.

Leo: Ooh, how exciting.

Steve: Yup.

Leo: Thank god that novel was no longer.

Steve: Only took me 10 days of furious reading. Believe me, I got it out of my system. I'm having...

Leo: I just bought the two that you mentioned on Audible.

Steve: Yeah, the Greg Mandel series, they are really - they're easier and really fun. They're sort of like Hamilton getting warmed up.

Leo: I can't wait. I'm so excited. But if you have questions for Steve, GRC.com/feedback is the place to go.

Steve: Yes.

Leo: Don't email him. You can't. GRC.com/feedback. And we will do a PGP, I hope a PGP episode soon.

Steve: Yes, well, we're going to do a series.

Leo: Yeah.

Steve: Because there's a lot of foundation I want to lay down.

Leo: A lot of questions. And I'm getting a lot of emails from people who have installed it and got it working, but there's still lots of questions. In fact, I will forward you an email I got from somebody who said, "You're doing it wrong, Leo." And you should have multiple keys. You should have a different key for encryption, a different key for signing. You shouldn't be using RSA, et cetera, et cetera, et cetera. So I'll pass that along to you. I guess this is the pro tips for PGP.

Steve: Wow, cool.

Leo: Yeah. I mean, I've been perfectly happy with the old way, but if you really wanted to make it breach-proof, I guess...

Steve: Yeah, when you're just saying "Mom, what's for dinner..."

Leo: Yeah. There's nothing in there. But I liked it. I just like the tweak the NSA as long as I can, till I'm in jail. What else? I guess that's it. We will see you next week right here.

Steve: Perfect.

Leo: Thanks, Steve.

Steve: Thanks, Leo.