The TrueCrypt Audit

Description: Leo and I catch up on a busy and interesting week of security events. Then we take a close look at the results of the just-completed second phase of the TrueCrypt audit, which focused upon the implementation of TrueCrypt's security and privacy guarantees.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-502.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-502-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. How long ago was it that the TrueCrypt audit was begun? It's finally done. Steve will tell us the good and the bad news next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 502, recorded Tuesday, April 7th, 2015: The TrueCrypt Audit.

It's time for Security Now!, the show where we protect your security and privacy online, 502 episodes of secure insanity with Mr. Steve Gibson. Hi, Steve.

Steve Gibson: And no end in sight.

Leo: No end - that's the good news. Things are not getting better.

Steve: Yeah, we had a busy week. Google threw their weight around with China NIC that we'll talk about. Firefox has jumped forward with a really interesting idea known as "opportunistic encryption," to allow HTTP sites that don't have strong certificates to still encrypt. Google had another mistake where on Saturday a critical certificate of theirs expired. And so that's part of the news. The other part is what they replaced it with, which I found interesting. Microsoft, there were announcements that Microsoft was abandoning Do Not Track, the DNT header. I'm not sure that's true, so we'll take a look at that. I found an interesting little blurb, actually someone tweeted it to me, I thought it was interesting, about the market in IPv4 space heating up as it becomes increasingly rarefied.

And the big news of the week, for our audience, at least, is that TrueCrypt, the second

phase of TrueCrypt audit was completed. And I've got Matt's short version from his blog, and an analysis of the full audit and what it means. So I think a great podcast today. Lots to talk about.

Leo: Matt is, of course, Matthew Green from Johns Hopkins, the cryptographer who's doing the audit. I cannot wait to hear. This isn't going to solve the mystery of what happened to the TrueCrypt programmer, but it's at least going to let us know whether we should be able to use TrueCrypt.

Steve: You know, yeah. What it gives us is two things that I think are really nice. It gives people who are using it and want to keep using it an exact calibration on that. And to the degree that there have been forks of the source, and we know there have, it also gives those people who forked the source some specific things that they could look at and improve on. So really useful, I mean, just across the board. So this ends up improving the future and, in my feeling, giving people a calibration on where they are now, if they're TrueCrypt users.

Leo: Well, I think there was some concern because when the guy abandoned the project he said, "Don't use TrueCrypt, it's not safe."

Steve: Yeah. And you know what my take was.

Leo: Yes. And I think...

Steve: I put up an archive and said, "Here it is."

Leo: It is safe. So we'll find out. Was Steve right? Coming up. What a tease. All right. Let's start with the security news, Steve.

Steve: Yeah. So, okay. So probably something maybe not surprising, but still certainly another example of Google throwing its weight around, was the upshot to that certificate that was discovered installed in MCS Holdings' proxy. Remember that somebody within MCS Holdings' network made the mistake of using Chrome to try to go to a Google property. Chrome has all of the valid certificates pinned, that is, essentially "pinned" is the cryptographic term. When you "pin" a certificate, you're no longer relying upon its signature, that is, that it is signed by someone you trust, sort of a one-degree-removed trust. You have pinned its hash, essentially, the actual entity of the certificate, and that's what you're checking against. Which means it cannot be spoofed. You cannot give somebody a certificate that looks like a Google certificate, that appears to be signed validly, but was signed by someone who is not actually trusted.

So Chrome comes with the knowledge of Google's valid certificate. So the second it sees something that appears to come from Google, but isn't, it sends alarms out. So that alerted, immediately alerted Google on March 20th, just last month, to this invalid certificate that they tracked down to CNNIC, China NIC, essentially, a major Chinese certificate authority. So the other shoe dropped on April Fools Day, April 1st, but this was no joke. Adam Langley updated his existing blog posting, saying: "As a result of a joint

investigation of the events surrounding this incident by Google and CNNIC, we have decided that the CNNIC Root and EV CAs will no longer be recognized in Google products."

Leo: Wow.

Steve: So this is removing the root trust of every certificate that that certificate authority, CNNIC, has issued. Because basically, whatever happened, CNNIC was unable to make Adam and company happy.

Leo: CNNIC said it's not our fault, somebody did this. Right?

Steve: Well, okay, yeah. So we'll get to that.

Leo: Okay.

Steve: So Adam says: "This will take effect in a future Chrome update." Because, again, they have to, like, essentially build this intolerance into Chrome. They have to remove a certificate from the root that Chrome brings with it because it has its own certificate technology. So he says: "This will take effect in a future Chrome update. To assist customers affected by this decision," which is everyone who has a currently valid certificate issued by CNNIC, "for a limited time we will allow CNNIC's existing certificates to continue to be marked as trusted in Chrome through the use of a publicly disclosed whitelist."

So, I mean, this is a mess. But that's what they say they're going to do. "While neither we nor CNNIC believe any further unauthorized digital certificates have been issued, nor do we believe the misissued certificates were used outside the limited scope of MCS Holdings' test network, CNNIC will be working to prevent any future incidents. CNNIC will implement Certificate Transparency for all of their certificates prior to any request for reinclusion. We applaud CNNIC on their proactive steps and welcome them to reapply once suitable technical and procedural controls are in place."

So that paints a very happy picture. We're all one big happy Internet community. However, CNNIC doesn't really see it that way. CNNIC, and I'm looking at a - oh, it's on 4/02 of 2015. I can see that date buried in the URL. So the day after this announcement, CNNIC posts an announcement on their own site, and it reads: "The decision that Google has made is unacceptable and unintelligible to CNNIC, and meanwhile CNNIC sincerely urge that Google would take users' rights and interests into full consideration. For the users that CNNIC has already issued the certificates to, we guarantee that your lawful rights and interests will not be affected." And it was signed China Internet Network Information Center, April 2nd, 2015.

So this is all we know. We don't know anything more about what was going on behind the scenes, what motivated Google to do this. It's got to be more than we're seeing because this is big. I mean, this is, first of all, it's not just removing the root, but it's coming up with some way of whitelisting existing certificates, but that proactively then prevents any non-whitelisted certificates. So maybe CNNIC will provide Google with a whitelist, and so that's what CNNIC means when they're assuring their existing

customers that their service will be noninterrupted. Now, of course the other question is, are the other browsers going to follow suit? Are we going to see Mozilla and Microsoft and Apple yank the cert also?

Leo: They kind of have to. I mean, it's not meaningful if they don't; right?

Steve: Well, yeah, I mean, or it just could be Google being, you know, strong-arming again.

Leo: Of course they can then say, "Hey, we're the safest browser," if the other guys don't.

Steve: And one thing it does is, if the Chrome - so the idea, what would happen is Chrome would not be able to visit any large CNNIC website customers. That is, big businesses, commercial websites, any entities who had a certificate from CNNIC in the same way that I got mine from DigiCert.

Leo: Well, wait a minute. You could go visit it, but you'd have to override the thing that says don't go here, this cert isn't good.

Steve: Maybe. See, you cannot do that on sites that are enforcing SSL. So, for example, my site...

Leo: Oh, interesting.

Steve: ...which uses HSTS, one of the benefits of that is we really up the ante. And if a site is offering Strict Transport Security, the browser will not allow you to bypass that. And also, I mean, if you want to log in, and you don't have - and Google, if you want to log in on Chrome, won't accept the SSL cert. Even if you did force it, you'd be doing an in-the-clear login. Your credentials, I mean, you know, your credit card information, nothing would be encrypted. So it's, I mean, it's a disaster. So it's affecting all of the websites who purchased their certificate from CNNIC.

And whatever it is that happened, Google was unsatisfied with CNNIC's response, maybe with their procedures, with their auditing, I mean, who knows what. We just don't know more. But it's got to be serious for Chrome to just say we are removing a major certificate authority from the trust store in Chrome until such time as they, I mean, Adam says, "We applaud CNNIC on their proactive steps and welcome them to reapply." Yeah, reapply for global trust on the Internet because Google feels that they [crosstalk].

Leo: Google needs to be transparent, though, about the reasons they did this. I think it's unacceptable for them to just do it and not say why.

Steve: Yeah. I mean, and Adam's posting is all peaches and cream. It's like, oh, well, we had...

Leo: That's typical Adam Langley. It just drives me nuts. I mean, maybe they're - I'm sure they have a good reason. If they do, they need to say what it is, though. Otherwise, this is, you know, kind of like a trial without representation. It's a kangaroo court.

Steve: Yeah. I mean, all he says, "As a result of a joint investigation," blah blah blah, "we have decided that the CNNIC root and EV CAs will no longer be recognized in Google products. That's phenomenal. I mean, it puts them out of business, essentially. Just, bang, you're out of business because Google now...

Leo: When they say "joint," them and who?

Steve: Yeah, well, exactly.

Leo: Joint with whom?

Steve: "As a result of a joint investigation of the events surrounding this incident." Oh, by Google and CNNIC.

Leo: Okay. So CNNIC investigated. It's very confusing.

Steve: Yeah. And CNNIC's position, which we talked about before, is that they issued an intermediate certificate with signing privileges to MCS Holdings under the understanding, the agreement that MCS Holdings would only sign certificates for their own domains. But when MCS Holdings instead put that certificate in a proxy, it was signing certificates for every domain that went through the proxy. And Google's was one of those. It synthesized, it fraudulently created a Google certificate, which Chrome saw inside the proxy, and all hell broke loose.

Leo: Wow.

Steve: Because that's - Chrome is on a hair trigger.

Leo: That's amazing.

Steve: Yeah. But the consequence is this is the kind of - this is the response, I mean, and this is really - this is the strong side of Google that we're seeing is, oh, and, well, we'll get to this in a second because I was going to say people have been commenting and asking why GRC's using obsolete security, or obsolete cryptography. It's like, oh, boy, okay. So anyway, so I thought this was interesting. We'll see what happens with this and keep an eye on it. But it's like, here the majority browser now on the Internet just says we're no longer going to trust any certificates without some sort of whitelisting guarantee, but apparently nothing new that CNNIC signs will be so. So its existing

customers are being kind of grandfathered in, in a whitelist. Presumably CNNIC will provide that to Google, which will then - a public whitelist. So we'll all be able to see it. And maybe that's so that the other browsers can do it, too. Maybe Google is going to promote this across the browser industry. Wow.

Leo: You know, this stuff is complicated. We had a listener send us a note saying, "I can't download from your podcasts anymore. Something changed at work, and now it's saying that Cachefly, who's our provider, is using SSLv2, and so work said we won't download it." But what's weird to us is, first of all, Cachefly says no, we're not. And it's not HTTPS. So what we think is somebody's using HTTPS Everywhere. And there are risks to enforcing HTTPS on sites that don't do HTTPS.

Steve: Yes, yes.

Leo: But we can't figure it out. But I think that that's what's - we think that's what's going on.

Steve: That makes sense, that some crazy IT guy said, okay, we're just going to not have - we're not going to allow non-HTTPS connections. So just - and this is the problem. There are still services where, like, a CDN. What does a CDN need security for, delivering MP3s?

Leo: Right.

Steve: Ultimately, yes, that's where we're going to move. But let them do it when they want to.

Leo: He wrote, "I and several of my friends can no longer access any of our favorite TWiT shows. This happened when our tech security rep removed SSLv3 from our browsers. If you have any influence with Cachefly, it'd be nice if they stopped using SSLv3." But they're not. So you're right, they did add something else. They added a policy that said...

Steve: Wow.

Leo: Or maybe they just turned off podcasts. It is work, after all.

Steve: Yeah. Yeah, exactly. Someone looked at the size of those downloads and said, wait a minute, you know?

Leo: Just don't, I mean, don't turn on HTTPS for Cachefly, I guess, is the key.

Steve: Yeah. So we're going to talk about Firefox for a moment and then come back to

Google, just because I don't know why.

Leo: Because there's so much to say.

Steve: Because there is. So Firefox began to offer something known as OE TLS, Opportunistic Encryption TLS. Opportunistic encryption is an interesting idea. Everyone who's been listening to the podcast certainly understands the way the Certificate Authority system works, the way I purchase a certificate from my CA, DigiCert, every few years. They recertify me, make sure I'm still here, phone calls, email, go through that process. In return, they sign my certificate asserting my identity so that somebody else who goes to GRC.com is receiving that assertion signed by someone they trust. In this case, they trust DigiCert, who has down their homework.

But that's the process for authentication and encryption. There's a case to be made for encryption only, that is, you know, if we just have HTTP, well, we have neither authentication nor encryption. Everything's just in the clear. So session cookies, like Firesheep was able to sniff and do session hijacking. Those are in the clear. With HTTP, for example, you could have a man in the middle who was intercepting and modifying things. I mean, there's no protection at all. But it's easy because it requires no certificate, no authentication. Up until recently it was faster than bringing up a secure tunnel and so forth.

So that's the way things had been. Not all sites, the gurus of the Internet argue, really do need authentication. That is, sites that are not using it, are not using any encryption now, could at least add encryption so that, well, I guess I would call it "opportunistic sniffing." You know, just for example, we think that the NSA has big taps all over the Internet, and they're just spooling everything that goes by into their server farm in Utah to look at it later. Now, if it's encrypted, that's a lot more work. But to the degree that anything is still not encrypted, it's in the clear. And we know from, like, sniffing hotel WiFi and non-switched networks, there's a huge amount of passwords and usernames and things still passing by in the clear.

So opportunistic encryption is a means for allowing a website to sign its own certificate, so-called "self-signed certificate." So a website would offer a certificate which allows the standard SSL/TLS handshake to proceed, with the understanding that no higher authority has verified that website's identity. It's the website asserting its own identity on its own behalf. So it would be xyz.com makes their own certificate, and they sign it. And so, I mean, and that's possible. In fact, that is exactly what the root certificates are. The root certificates that our certificate authorities have installed in all of our devices, those roots are self-signed. They sign themselves as the anchor. And the only reason we trust them is that they came with the operating system. They're in a protected store. And so we look to them for the signatures of all other certificates. So with v37, which is just out, Firefox has added this facility.

Now, the problem is, in order to do it, it's kind of weird because you can't - a website can't just put a self-signed cert on a regular domain where anyone could connect to it because no browser's going to trust it. If you just connect to it on port 443, like https://xyz.com, well, your browser's going to say, what the heck is this? You sure you want to do this? I mean, all kinds of bells and whistles would go off because your browser would see that it was being given a self-signed certificate.

So the way this works is you go to http://, that is, standard HTTP domain. And the browser says, oh, fine, you know, xyz.com, no security. It connects on port 80, as it

normally would. But when it gets the server's headers, the response headers, from the first reply from the server, if this server wants to support opportunistic encryption, there's an additional header that's added to the response. It's Alt-Svc, alternative service, alt service. And so it's Alt-Svc:, and then either "h2" if the server is supporting HTTP/2 on some other port, or "spdy." SPDY could also be used if it was a server that didn't yet support HTTP/2, but did support SPDY 3.1, for example.

So that header says there's an alternative source of the same service on that port. And typically it is also 443, just so you can get through proxies and other filters and things on the web. You probably don't want to go off to some weird port because you might find other things would block you. So what then happens is Firefox continues using the standard connection while in the background attempting to negotiate and bring up a TLS connection on the specified port, because that Alt-Svc header tells the browser what port to go to and which protocol to use, h2 for HTTP/2 or spdy. If the browser is able to succeed, it then preferentially switches to using that protocol for all subsequent traffic.

And the one other little addition is the browser will cache that information. It'll cache the fact that a given domain, a given website was using an alternative service. And at some point in the future - and there's control over the cache, the length of time that the cache will remember that. If the user goes back there just using http://, the browser will remember that that server was offering TLS on some other port and try it first. So it'll bring that up from the beginning.

So it'll be interesting to see if this catches on. To me it's like, well, okay. Seems like you're going through a lot in order to have sort of a soft switchover, sort of a handoff to encryption. You can't rely on it because you're not encrypted initially. If the connection won't come up, then the browser doesn't switch. It continues to use port 80 and just HTTP. Which means bad guys could easily interfere. If they simply blocked the incoming traffic to that website, then, that was offering opportunistic encryption, nobody would use it because it would just be blocked. And so everybody would say, oh, you know, the Firefox browsers trying to do this would say, oh, I guess that server's down at the moment, and continue using port 80. And people could block it even after you had a session because it turns out that, if at any point the connection goes down, Firefox falls back to port 80 again.

So after all of this, Mozilla brought it out in 37. And within hours a problem was found. A security researcher very cleverly, probably just trying to play with this, realized that, if the alternative service also had an alternative service header, you could redirect traffic to somewhere else. So it was like, oops. And Firefox immediately - it was an easy problem to fix. It was just something that they had overlooked. They had Firefox processing Alt-Svc headers, not only on the unencrypted connection, but on the alternative service connection, and that led to some chaining possibility. So 37.0.1 was released like a day later, and we have that now.

So anyway, I wanted to let everyone know that exists. I'll be surprised if it ends up being a big deal, if sites jump on it, because you can get free certs, legitimately trusted certs for free that last a year from a number, like, what, StartSSL, I think, is one of them. So there are solutions for this, if you want encryption, that also gives you authentication and just a more straightforward way of doing it, rather than this weird sort of kind of hybrid kludge.

Now, back to that picture at the top on the first page of the show notes, Leo.

Leo: Yes. I realized I showed it prematurely. I apologize.

Steve: That's all right. So this caught Google by surprise on April 4th, which was Saturday morning. A major service of Google, if you look at the name on the top cert there, the common name...

Leo: Oh, Gmail. The outbound server for Gmail.

Steve: Smtp.gmail.com expired. Now, what really happened - yes. What really happened was, well, and that's what happened was suddenly nobody - everybody was getting errors on trying to use the SMTP service at Gmail.com because they were no longer able to get secure connections. Everyone's servers and clients were coming back saying, uh, this certificate is no good.

So if we look at this chain, and for those who aren't able to look at it, the CA at the root of the chain is GeoTrust. So GeoTrust Global CA, GeoTrust Inc., signed an intermediate certificate that they issued to Google, the so-called Google Internet Authority G2 cert. And that certificate was valid from April 5th of 2013 to April 4th of 2015. Yeah, 2013 to April 4th of 2015. So a two-year interval, one day less than two years. And so that's Google's intermediate certificate that they were then using to sign other things. We don't know what other things. But at least they signed this smtp.gmail.com cert, which was installed on the smtp.gmail.com server, and expired Saturday morning.

Now, the way this works is that someone's email server may not trust directly the Google Internet Authority, that is, the intermediate cert. So the certificate that is from the mail server bundles in, it includes the certificate that signed the end certificate because that intermediate certificate is signed by GeoTrust, and everybody will trust that. So the point is that the way this certificate expired was it wasn't really the end certificate. That expired, will expire, would have expired on December 30th of 2015, the end of this year. But the one whose trust it was dependent on, the intermediate, expired Saturday.

Now, Google, it took them about, like, eight hours, I think it was, to get this thing. I mean, it caught them by surprise. They had to figure out where the problem was. They had to essentially reissue, get a certificate reissued that did not have an expired intermediate. And they may have had, like, lots of other certificates with intermediate certificates with later expiration dates, and somehow the order in which this happened just sort of caught them by surprise.

Further down in the notes I show the new certificate chain, which is current as of today. And so the root is still GeoTrust. Now the intermediate authority is valid through December 31st of 2016. So whereas that expired on April 4th, 2015, now it's good for the rest of this year and all of next year. And then it signs the end certificate, that doesn't look like it's changed much. It's still expiring at the end of this year. But what I just sort of looked at and shook my head was the signature algorithm that every one of these newly issued certificates is using.

Leo: Oh, no, no, no, no.

Steve: Minted on Saturday.

Leo: They're still using it.

Steve: Signed with SHA-1. Of course they are. And you know why?

Leo: Why?

Steve: Because the browser won't complain.

Leo: But they won't let us do that.

Steve: Correct. Now go to GRC.com, Leo.

Leo: Right now?

Steve: Yeah.

Leo: Okay. I'll do it in Chrome.

Steve: GRC.com. Bring up - oh, I'm sorry, yes, you have to. I should have specified. Because nobody else cares. Everybody else is happy.

Leo: Yeah. You look good on this. You're green. You've got the green cert.

Steve: But click on the greenness.

Leo: All right.

Steve: And go to that second tab there.

Leo: Connection.

Steve: Yeah. And what does that say there, the second item?

Leo: Obsolete cryptography. Steve, I'm shocked. Shocked.

Steve: Yes. So Chrome is saying that my site is using obsolete cryptography, despite the fact that you'll notice it's TLS 1.2, the latest standard. They're bitching that I'm using...

Leo: It's the SHA-1; right?

Steve: SHA-1, yes. And they are, too. Newly minted certificates protecting Gmail are being signed, being now created with SHA-1. Just because no one is going to complain.

Leo: Bugs me a little bit. Yeah, bugs me a little bit.

Steve: I know, I know. And in fact, I've heard from people within the certificate community, the CA world, that this is causing havoc. A person who works at a CA said to me the other day, "Steve, if I could tell you what a mess Google is making by scaring people." And in fact someone tweeted me just this morning a beta of Chrome with https in red and a red slash through it, which is now the way my site appears in a forthcoming version of Chrome. So, yup, Google...

Leo: Oh, because they're escalating it; right? Yeah.

Steve: That was the plan. They're escalating. And actually mine may not because my certs all expire at midnight on New Year's Eve. Somebody said to me, oh, by the way, Steve, don't forget, you're going to be at the TWiT Brick House on midnight of New Year's Eve.

Leo: Oh, not this year. We're not doing it this year, so you're safe.

Steve: Oh, good, perfect. I'll be home.

Leo: We know where Steve will be.

Steve: I'll be home to change my certificates. There will be a countdown. As the ball is dropping in New York, I will be - actually, I have to synchronize.

Leo: You could do it before then; right? I mean, you don't have to...

Steve: Oh, of course I can, yeah. And I'm sure I will. But it just - it did bother me, this double standard that here Google is telling everyone that SHA-1 certificates are obsolete, and they're implying that they are insecure. Yet they're using them still where no one can see them because they know they're just fine. Gmail is protected with SHA-1, just on Saturday.

Leo: Kind of frustrating.

Steve: So, yeah. So Microsoft. I see, like, news blurbs saying Microsoft is abandoning Do

Not Track. Now, and you of course could do that much better with your Walter Cronkite.

Leo: Microsoft now abandoning Do Not Track.

Steve: Perfect.

Leo: Tracking news.

Steve: So on April 3rd - which I guess, if the 4th was Saturday, when the Google certificate expired, the 3rd would be Friday - they blogged about some change. Now, I'm not going to paraphrase, I'm going to read this, because I don't know what this means. And I guess we won't know until we see it. But here's what they said, and then we'll talk about it.

Microsoft writes: "As industry standards evolve, how we implement those standards evolve as well. So to reflect the current requirements of the privacy standard for tracking preferences, Microsoft is changing how Do Not Track (DNT) is implemented in future versions of our browsers: We will no longer enable it as the default state in Windows Express Settings. While our implementation of DNT two years ago in Internet Explorer 10 was welcomed by many, others voiced concerns, especially given that discussions were underway at the time to establish an industry-wide standard for user tracking preferences.

"Since then, the World Wide Web Consortium (W3C) has continued to refine language to address how users express a preference regarding tracking. The latest draft of the standard reads: 'Key to that notion of expression is that the signal sent MUST reflect the user's preference, not the choice of some vendor, institution, site, or network-imposed mechanism outside the user's control. This applies equally to both the general preference and exceptions. The basic principle is that a tracking preference expression is only transmitted when it reflects a deliberate choice by the user. ? In the absence of user choice, there is no tracking preference expressed.'"

So Microsoft resumes: "Put simply, we are updating our approach to DNT to eliminate any misunderstanding about whether our chosen implementation will comply with the forthcoming W3C standard. Without this change, websites that receive a DNT signal from the new browsers could argue that it doesn't reflect the users' preference, and therefore may choose not to honor it. As a result, DNT will not be the default state in Windows Express Settings moving forward; but we will provide customers with clear information on how to turn this feature on in the browser settings, should they wish to do so. This change will apply when customers set up a new PC for the first time, as well as when they upgrade from a previous version of Windows or Internet Explorer."

So I don't know quite what Microsoft means when they say "This change will apply." I guess they mean the lack of future defaulting it to on. "We said in 2012 that browser vendors should clearly communicate to consumers whether the DNT signal is turned off or on, and make it easy for them to change the setting. We did that for IE10 and IE11. And we're continuing to do so with future versions of our browsers." So I guess my reading of this is they got a lot of heat, and they're backing off.

Leo: Interesting.

Steve: Yeah. Yeah, so...

Leo: This is the same heat everybody else is getting. I mean, this is an interesting - there's two different parties to this, on both sides; right?

Steve: Right. The dilemma is that we have the so-called "tyranny of the default," you know, the expression I coined. I like it because it just sort of says most users just leave things the way they are. Whatever the default is, that's the way they're going to be. So Microsoft, with some fanfare a couple years ago, and we talked about it on the show, decided that the express settings for the browser, which the user would just say, you know, kind of look through the list, do you want it this way, and the user would say, "Yeah, that all looks right." Enabling Do Not Track was there. And understand that this argument about the signal must be set and must represent a user preference, we talked about this two years ago. This isn't new. This isn't something that just happened.

So I don't think anything has changed except something has happened over on Microsoft's side to, say, uh, you know, this, I mean, these arguments, the idea that, if it was on, then websites could say, well, that really doesn't represent a user preference because Microsoft turns it on by default. So we're going to choose to ignore it in that instance. So we'll have to see how Microsoft presents this. If they don't produce a neutral question, then it's certainly the case that they have abandoned Do Not Track. If the user has to go in search of it, then that would constitute abandonment. If, however, Microsoft says, "Hi there, no bias shown, do you want to be tracked or not," then that would represent a user choice that would be explicit.

Leo: Yeah. And of course they don't like to put too many of those up as you sign up for these things. They don't, you know...

Steve: Right, it overwhelms everybody.

Leo: Yeah, yeah.

Steve: So, interesting. So this is an interesting pair of charts. This is two charts that show the rate of IP address block movement and IP addresses, individual IP addresses over the last couple of years. And so they're both basically exponential curves, going up crazy. For example, in November of last year, toward the end of last year, in that one month - or is that a week? November, oh, yeah, that looks like, no, that's probably one - oh, yeah, I'm sorry, it's one month because there's 12 bars per year. So somewhere like November 2014, nearly two million IP addresses transferred.

Now, that means that there were IP addresses that were being routed down some channel, some tube, one of those Internet tubes, going in some direction, and something happened to cause one or more typically large blocks of those IP addresses to go now be routed somewhere else. Meaning that someone gave them up, and somebody else got them. So they're being transferred. It turns out that there's an actual sort of a - there's

like an aftermarket now that is developing for IPv4 space, even though IPv6 exists, and it's got enough IP addresses that we could all have our own Internet worth several times over. I mean, when I talked to my provider about getting some IPv6, they said, "So what do you want? You want 64,000 IPs?" It's like, what? It's just me here.

Leo: They're free. They're plentiful. No reason not to.

Steve: Yeah. So what's happening is there is now a resale market. And so, for example, it turns out that for whatever reason, a fluke of history, Romania has had a surplus of IPv4 space. And Jump.ro has been selling it off like crazy. In fact, 51 percent of all the IPv4 blocks transferred last year came from that one Romanian organization, Jump.ro. They will sell large blocks of IPv4 space for $10 per address, or lease smaller blocks for 50 cents per address per year.

And to give you some sense for this, of like where they're going, Saudi Telecom purchased 1.5 million IP addresses in 17 block transfers, 14 of those from Romania, three from Ukraine. So if we sort of back-of-the-envelope say $10 each, they paid $15 million, Saudi Telecom did last year for 1.5 million IPv4 addresses. So it's not surprising. As we know, change is hard. There's a huge install base, well, I mean, all of the equipment on the Internet knows IPv4. IPv6 is still moving forward, but at a glacial pace. And it's just easier for people who need more existing IP space to find people who have it and buy it from them. So that's happening. It's a strange world, the fact that, you know, it's like a virtual currency. It's like, "You got any IPv4? What can you part with? If you're not using it, we'll pay you for it." And of course...

Leo: Funny.

Steve: And one thing that this does is it tends to cause routers problems because what we really want, the original fabulous architecture was that IP space, the IP protocol, the addressing, was hierarchical. That's why, for example, HP started with 14. Meaning HP was everything 14. And so HP could basically, if an IP address began with 14, all the router had to do was send it to HP. Didn't even have to look at the other three bytes, just, oh, 14, HP. And many of the early networks were like that. GE has a Class A network, as those are called, and many major organizations. Many universities are still today squatting on - maybe they're waiting for the price to go up. Once it hits a hundred dollars for IP, it's like, ah, we can lower tuition and sell off some of our bulk IPv4 space.

So the problem, though, is, as the IPv4 space becomes more fragmented, that is, for example, while Romania had a huge block, all of that, like one table entry in the router could match a huge number of IP addresses underneath their network number and just send everything to there. But now, with Romania selling these off to different parties, what was one entry in all of the routers, all of the big iron routers that do this on the Internet, what was one entry now might be a hundred entries because, if they've sold these pieces of one big block, scattered them all over the place, now we need individual address matches on the routers in order to send them off to different destinations. So this is causing concern among the engineers who worry about the Internet's plumbing because, due to this kind of fragmentation, there is a routing cost that the original engineers just brilliantly sort of considered and came up with an economical solution to in the beginning.

So, miscellaneous stuff. I ran across an amazing add-in. We've talked about Firefox

configuration, you know, how you do about:config, and up comes so many individual entries that, forget about it, you have to have a search bar. And so you put in, like, UDP or SSDP or OCSP, for example. And then it filters out this galaxy of individual tweakable things to just the ones that you hopefully care about.

Turns out somebody built a UI - and I know that we've got a bunch of tweakers here in our midst on the podcast, and people who are, as I am, still using Firefox because it's so much leaner than Chrome is - called Configuration Mania. So it's a Firefox add-on. I imagine you can just go to add-ons and search "configuration mania." It's -4420 is the number in addition to the name for the Firefox add-on. It gives you basically a beautiful, tabbed, multidimensional, laid-out user interface for that about:config. More settings than you, I mean, believe me, this is - you will get lost if you're someone who likes to play with things because it exposes them all, just in a very easily browsable fashion. So I wanted to provide that tip for our Firefox users who are also enjoying tweaking things.

I got a kick out of a tweet that I received last week, following up on our discussion, Leo, you and me, about SETI and Active SETI and the fact that we've been listening with all of these big ears pointed at the sky for all this time, and all we hear is just white noise. And so Barry Wallis tweeted, he said, "Steve, aliens are smart, so of course all their comms are encrypted and indistinguishable from noise, just exactly like what SETI sees every day." And it's like, of course. They're not going to be sending out - they're going to have some crazy...

Leo: Gee, that's a good point, yeah.

Steve: Yeah, it's going to be encrypted, and it's going to be compressed.

Leo: Yeah, it wouldn't be plaintext.

Steve: No. So we now are sending up...

Leo: So there'll be no patterns.

Steve: Right. It's noise. And, gee, so as far as we know, we've been receiving encrypted alien communications since we started listening.

Leo: A very good point.

Steve: We just don't know what the password...

Leo: Any sufficiently advanced civilization is going to send all its communications encrypted.

Steve: So, Leo, we don't know the password. I don't think it's "monkey."

Leo: Is it "monkey"? But they don't have monkeys.

Steve: They may not have come down out of the trees. Maybe it's "lizard." Maybe they're reptilian, you know, because apparently some aliens are. So if anyone has any ideas for the aliens' password that then SETI would be able to use to decrypt all these communications, that would be handy.

And equally strange, Indiegogo has a project underway to allow people to purchase a saliva test to measure the average length of their telomeres.

Leo: Wow. And why would I care how long my telomere is, whatever that is?

Steve: Okay. Well, the telomere, you'll know what it is, I'm sure you've heard about them. They're the little bits of protein on the ends of our chromosomes that essentially keep them from unraveling. They're likened to the little plastic tips on our shoelaces.

Leo: The aglets.

Steve: Yes. And so these are genetic aglets. And telomeres limit the number of times cells can reproduce. And so, for example, there's an enzyme called "telomerase" which is the enzyme that lengthens existing telomeres. And they are shortened when cells divide. The point is, they've been linked to health, in the same way that inflammation and other sort of underlying factors have. So of course you and I both did our genetic tests at 23andMe. I signed up for this little $89 saliva test, just because I'm curious. I mean, it's not clear that we have much control over it. There actually is a ridiculously expensive supplement, it's like $600 a bottle or something, that has been shown to lengthen someone's telomeres. I don't think I want to mess with this because...

Leo: We don't know everything there is to know about telomeres.

Steve: Mother Nature often has a good reason. For example, cancer generates telomerase in order to get the accelerated tumor reproduction that it needs. So it's figured out how to bypass the cell division limitation that telomeres create by sort of building its own accelerator. So it's like, yeah. But anyway, I'm curious. And I just thought I'd give a shout-out for it to anyone who might also be interested. I think they're about halfway through their campaign, about another month left to go. So it's at Indiegogo. It's Titanovo, T-I-T-A-N-O-V-O, Measure Your Health. I imagine if you google "titanovo measure your health" you'll be able to find them. So, and I'm just curious. I don't know what mine are going to be. But, you know, so I want to find out.

Leo: I bet you have massive telomeres.

Steve: I have a feeling mine are in good shape. I've - yeah, yeah. Those indications are interesting.

Leo: Is it like midichlorians? I mean, is it, like, maybe - who knows.

Steve: Yeah, it's different, yeah. Just it's another parameter that...

Leo: I don't like knowing something that I can't do anything about.

Steve: Yeah, it's funny because there's one called CRP. C-reactive protein is a systemic measure of inflammation. And when I was doing my annual physical, and I've had mine measured, mine's...

Leo: You can get a CRP test, yeah, yeah.

Steve: Oh, yeah, yeah. But I remember I asked my doctor to add it, and he kind of looked at me, like, why? I said, "Well, because I want to know." He says, "Well, you can't do anything about it." And I said, "Well, actually you can." And that's why mine is right down at, like, 0.3 at the moment, just because there are things you can do to keep track of it. And it's weird, too, because I once had a really bad infection, didn't even think about it, this is maybe 10 years ago, when I had a CRP test done, and it was off the scale. It was like, 12. It's like, hey, it works, because I had never seen it do anything before. It was just always sort of down there like water. But it turns out, yeah, it actually does measure that. So, yeah, I guess I'm interested sort of as a point of departure for research. So, you know, I'll find out what mine is.

Leo: Good.

Steve: So a nice note from a - I sort of liked this one because this dates - I don't know how far back the 1980s are. What is that, like 30 years?

Leo: Long damn time.

Steve: A contemporary Security Now! listener, Paul Windham, recounts his first encounter with SpinRite. He said, "Dear Steve: As a contemporary SpinRite user I have to tell you about an experience [stammering], an experience I had with SpinRite many years ago." Sorry. "I was in the Marine Corps in 29 Palms, California in the..."

Leo: Nice, nice berthing there, that's good.

Steve: Yeah, "...in the 1980s. We had a PC which was running some kind of MS-DOS and would no longer boot. I was fortunate enough to work with a retired Marine named James Kornegay. He helped me all the time with PC issues, and he owned a copy of SpinRite, which we ran on the unbootable system - after which, of course, it booted right up and ran. To this day I am a SpinRite user, and it has saved me from a lot of pain. Thanks for 500 shows and look forward to odometer reading 512. Paul Windham,

SpinRite user, Security Now! user." So, Paul, thanks for sharing. And 30 years of SpinRite use, that's great.

Leo: I'm sorry, I was playing with my phone. I got the new...

Steve: With the fabulous fingerprint reader, I know.

Leo: I got the new Galaxy S6, and I'm just - it's so pretty, I'm just caressing it a lot.

Steve: Ah, nice. Small.

Leo: Well, it's not as big as the Note, you're right.

Steve: I'm just teasing.

Leo: It's a five-inch phone.

Steve: I'm just teasing.

Leo: No, you're not. You're making me feel bad.

Steve: And wasn't there a note about a bigger Apple phone coming? Was that just rumor? I can't...

Leo: A bigger? No, that's - there's no - we don't know what Apple's going to do. September there'll be a new phone. That's all we know.

Steve: Boy, this Watch sure is creating a lot of stir.

Leo: Yeah, we can't stop talking about it.

Steve: Well, and just it sounds like it's going to turn the Apple Stores upside down. I mean, really just be a mess. I'm going to stay away for a while.

Leo: That's what they're saying. Don't go to the Apple Store anymore.

Steve: Yeah.

Leo: Stay home and order online.

Steve: Yeah.

Leo: So much of what we do, we could do in the cloud now, you know. There's no reason to go to a brick-and-mortar store.

Steve: Right.

Leo: Push a button, and a uniformed member of the United States government will come to your door with it.

Steve: And say, "Here you are, sir."

Leo: Here you are. Steve Gibberino, let's talk about the TrueCrypt audit.

Steve: So certainly our longstanding listeners will all know about that weird day when we woke up and discovered that the TrueCrypt website, which had been there patiently making available versions of TrueCrypt for years, just suddenly went belly-up, disappeared.

Leo: So unexpected.

Steve: Yeah, well, out of the blue. Nobody expected it. The authors, my theory is, they wanted to be done with it. They wanted to be through. And, I mean, not just we're not going to keep updating it any longer, but we want to be disconnected. And as I said at the time, the problem with disconnecting from TrueCrypt is that it was the most popular whole-disk and partition encryption system there was because it was beautifully designed, easy to use, really solid. I mean, it was what we all recommended for people to use. I've used it. My tech support guy Greg relies on it to encrypt his laptop.

And so for that kind of application, people - even if these guys said we don't want anything more to do with it, well, people are still using it. They're, like, well, but wait, you know, what about support? What about questions? What about new versions of Windows and so forth? So my theory has always been that what they chose to do was to paint it in the worst possible light in order to help migrate people away from it. So they said, "TrueCrypt may be insecure; and, if so, we're not going to fix it." I mean, and so for people who were made uncomfortable by that statement, they went in search of an alternative. And that's what the authors of TrueCrypt wanted. They didn't want to just have this thing stop. They wanted to reverse time. They wanted to be through.

So the problem was that this occurred in between the first phase and the second phase of the TrueCrypt audit, which was a project that had been started by and sort of run by a group of cryptographers, Matthew Green, whose name we know well, from John Hopkins and some others. Money was put together in order to fund an audit of TrueCrypt

because, since it was so popular, since this was the one that people were using, someone thought, hey, you know, just the fact that it's open source - and this is what we're realizing with things like the OpenSSL bug that was sitting there on the 'Net for two years before we discovered that problem. Just the fact that it's open source doesn't guarantee that it's secure.

Leo: Right.

Steve: The only way...

Leo: In fact, that's what surprised me about this whole thing is that I'd just assume, well, somebody's looking at it. But apparently nobody was looking at it.

Steve: Right.

Leo: Not in any rigorous way, anyway.

Steve: Right. And so it was like, okay, time to audit TrueCrypt. So the first phase had been finished and came out fine by the time that the TrueCrypt site shut down, and the authors said, "Stop using TrueCrypt. Oh, by the way, Microsoft makes BitLocker. If you're using Windows, go over there." So the first phase was finished. The first phase, though, wasn't the deep crypto audit. They just looked at sort of the startup of the system. They looked at the boot technology, at the way the system got going, and sort of like the whole, the front end of TrueCrypt to see how that operated, looking to see if there was anything that was in any way suspicious. And it came up perfect. Nothing was found that looked wrong that far.

So one of the questions that came up, because we all knew that there was a Phase 2 deep crypto audit looming, was whether these guys knew something that we didn't, and they were bailing before the results of the second phase of the audit came out. That was, again, one of the multiple theories of why they did this because they really didn't give us much reason. They just said, "We're not doing this anymore." No reason given, really. Although, as I said at the time, there was some Twitter chatter where someone was able to get one of the authors in a dialogue, and they just said, you know, we're done. We want to move on. And as a developer of software, I can and could and do understand that.

Leo: Oh, yeah.

Steve: So audit results are available. The Phase 2 audit is finished. I'm going to read Matthew Green's short summary. Then we're going to look at in detail the four things that the auditors found. So Matthew Green said: "The TL;DR is that, based on this audit, TrueCrypt appears to be a relatively well-designed piece of crypto software. The NCC" - that's the group that did the audit - "found no evidence of deliberate backdoors or any severe design flaws that will make the software insecure in most instances." And so it's those sort of qualifiers where we're going to spend the rest of this time because we're going to exactly qualify or quantify what those instances are.

"That doesn't mean TrueCrypt is perfect," continues Matt. "The auditors did find a few glitches and some incautious programming, leading to a couple of issues that could, in the right circumstances, cause TrueCrypt to give less assurance than we'd like it to. For example, the most significant issue in the TrueCrypt report is a finding related to the Windows version of TrueCrypt's random number generator which is responsible for generating the keys that encrypt TrueCrypt volumes. This is an important piece of code, since a predictable random number generator can spell disaster for the security of everything else in the system.

"The TrueCrypt developers implemented their random number generator based on a 1998 design by Peter Gutmann that uses an entropy pool to collect 'unpredictable'" - and Matt put that in quotes, we'll talk about that as one of the issues - "'unpredictable' values from various sources in the system, including the Windows CryptoAPI itself. A problem in TrueCrypt is that, in some extremely rare circumstances, the CryptoAPI can fail to properly initialize. When this happens, TrueCrypt should barf and catch fire. Instead, it silently accepts this failure and continues to generate keys."

Now, as we'll see, this is not as dire as Matt, I mean, what Matt wrote is true. But again, we need to quantify this. "This is not the end of the world, since the likelihood of such a failure is extremely low." And for other reasons we'll see. "Moreover," says Matt, "even if the Windows CryptoAPI does fail on your system, TrueCrypt still collects entropy from sources such as system pointers, mouse movements. These alternatives are probably good enough to protect you. But it's a bad design and should certainly be fixed in any TrueCrypt forks.

"In addition to the random number generator issues, the NCC auditors also noted some concerns about the resilience of TrueCrypt's AES" - that's of course the Rijndael cipher - "code to cache-timing attacks. This is probably not a concern unless you're performing encryption and decryption on a shared machine, or in an environment where the attacker can run code on your system, for example, in a sandbox, or potentially in the browser. Still, this points the way to future hardening of any projects that use TrueCrypt as a base.

"TrueCrypt," finishes Matt, "is a really unique piece of software. The loss of TrueCrypt's developers is keenly felt by a number of people who rely on full disk encryption to protect their data. With luck, the code will be carried on by others. We are hopeful that this review will provide some additional confidence in the code they're starting with."

So that was Matt's statement on what the audit showed. Now I'm switching to the, I think it's about a 13-page PDF that details it. I'm not going to go through this in detail. I'm just going to start reading the authors' findings summary, and then we'll switch to the four things. So they said: "During the engagement, CS" - which is the name for Cryptography Services - identified four issues, and none led to a complete bypass of confidentiality in common usage scenarios. The standard workflow of creating a volume and making use of it was reviewed, and no significant flaws were found that would impact it.

"The most severe finding relates to the use of the Windows API to generate random numbers for master encryption key material, among other things. While CS believes these calls will succeed" - now, this is important, too. While CS believes these calls, these Windows API crypto calls, "will succeed in all normal scenarios, at least one unusual scenario would cause the calls to fail and rely on poor sources of entropy. It is unclear in what additional situations they may fail.

"Additionally, CS identified that volume header decryption relies on improper integrity

checks to detect tampering, and that the method of mixing entropy of keyfiles was not cryptographically sound." We'll talk about both those things. And then, "Finally, CS identified several included AES implementations that may be vulnerable to cache-timing attacks. The most straightforward way to exploit this would be using native code, potentially delivered through NaCl in Chrome," meaning Chrome's native code technology that we were talking about because of the RowHammer attack and how they removed the ability for a cache flush to be invoked. And it turns out, that same mitigation pretty much rules Chrome out as a means for weakening the AES in TrueCrypt, which they say: "However, the simplest method of exploitation through that attack vector was recently closed off." Ah, so, yes, they're saying that Chrome got fixed.

So, four vulnerabilities that we're now going to talk about in detail. Okay. So now four things. These are the four things, in order of worst to least bad, that everyone's opinion who audited this ranks. So the first one is that there is this Windows API call that we heard everybody talking about. The call, the actual API text is CryptAcquireContext. So that's what a programmer writes when they want to invoke this. It's saying to Windows, give me a context, which is sort of like an object-oriented container, for future crypto operations. It's sort of a way of allocating a chunk of memory which Windows will then use on your behalf. You give it back this context whenever you're doing things, and Windows manipulates it and says, okay, yeah, I did that for you.

So what they said was that testing on Windows XP indicates that, if this is the first time a user has issued a call with the null container, what happens is the way TrueCrypt uses it, even though this is supposed to acquire a context, and normally you give it a pointer to the container that Windows will fill, you can also call it just saying, no, I don't need a context, that's not what I'm using you for. And in fact that is the way TrueCrypt called it, with a so-called "null container," which is the second parameter to this call. They say that the first call made to CryptAcquireContext will fail, while the second, which initializes a new keyset, will succeed. And they said a later version of Windows tested appears to succeed on the first call, but this was not thoroughly tested.

So what they found was that, in a rather obscure case - and you'll notice that Matthew said that, and these guys said it, but no one's told us yet what that is - Windows XP could fail that call. Then these guys went on and said, "While disturbing, this issue should not cause failure on common XP uses. However, this is not the correct method of calling CryptAcquireContext, and it may cause failure on uncommon Windows configurations."

So first takeaway is all of the forked code should fix this because why not fix it? This was a bug in the code that on Windows XP, in a rather obscure instance, this call could fail. Now, I'll explain why, even if it does, we don't care. But when does it fail? So in their exploit scenario explanation for this first of four bugs, they said: "A user creates a TrueCrypt volume on a company-managed machine. Because of the group policy settings in place at the organization, TrueCrypt is unable to open a handle to a cryptographic service provider and falls back to insecure sources of randomness, potentially enabling brute-force attacks on the master key." So in other words, if you had XP, and if you were in a corporate environment, and if that corporate environment XP machine was using group policy settings which prevented TrueCrypt from opening a handle to the cryptographic server provider, then this one source of entropy for key generation would fail.

Now, listeners will remember that I looked closely at the challenge of generating a large amount of entropy when I was doing what we called "entropy harvesting." In fact, I think we did a podcast called "Harvesting Entropy" here [SN-456], talking about the solution that I found. And our listeners will remember that the idea was that you use a vast array of different sources of entropy - counters, timing, unknown stuff. And, for example, I am

using the cryptographic API in SQRL. I'm calling it, not incorrectly, and adding what it provides to the pool.

But at the same time, I'm pulling from innumerable other sources. Very high-resolution timing. I'm, for example, using the instantaneous number of clocks which have occurred at 3.4 GHz, which comes from the chip, bubbling up through the cache queue and instructions, so nobody on the outside can know what it is. The point is that you always want multiple sources, all being sort of pooled together into one source. And this was that idea that Peter Gutmann came up with and talked about in 1998 as a way of building an entropy pool.

So TrueCrypt, very much like SQRL, my solution, TrueCrypt uses the value from the CryptoAPI as one of many sources. The other sources: 11 pointers to a variety of data structures. Process and thread IDs enumerated from the instantaneous snapshot of that system at that moment. Milliseconds since Windows was started. The process startup time. The cursor position. The time of the last input message. Flags indicating what types of messages happen to be in the message queue at the moment. The X and Y coordinates of the input caret and the mouse cursor. And remember that, during TrueCrypt volume creation, you're told to move the mouse around randomly in circles and crazy zigzag ways until you get tired.

And so it's sucking all of that in. Statistics regarding the current memory usage and the working set such as the load measured on the system between zero and a hundred, total physical memory, available virtual memory, minimum and maximum working set sizes. The creation, the user, and the kernel execution time of the current thread and process. Network management data, and physical hard drive performance history statistics.

So, okay. That gives you a sense for the quantity of entropy being collected. So, yeah, it would be nice to toss a value from the Windows CryptoAPI into that also. But if XP happened to have group policies in place that prevented it from doing that when you created a volume, oh, darn. You got everything else. And so the question is, is this sufficient entropy? So what's being criticized here is the fact that there was a case in which one of these many sources might fail to add its value to the pool. Good to know. The forks of TrueCrypt might as well include it. But I don't think anybody is in any danger from this. That's the worst of the problems that was found is that one.

The second worst, which they considered severe, but high difficulty to exploit, is the AES implementation being susceptible to cache-timing attacks. So what that's about is that, in order to accelerate the performance of AES - the Rijndael cipher, which is the one we like, very fast and bulletproof. But in a hard disk, a full hard disk encryption system, nothing is more important than speed. I mean, it's sitting there in the channel between your drive and your machine. Every file you read, every file you write, the paging file, everything, is going through this encryption and decryption. So you absolutely want that to be as fast as possible. In order to speed up AES, precomputed lookup tables are created, essentially so that it's possible to replace a lot of computation with just looking at a precomputed value.

The problem is these tables typically do not comfortably fit entirely within the CPU's cache. They're bigger than the cache size. And if they did fit in the cache, then while you were doing the encryption/decryption, already cached values would be found in the cache and wouldn't have to be looked up again. So it's when the values don't fit in the table, you have a so-called "cache miss." And very clever people are able to use this as a so-called side-channel attack, that is, say that they were running in a shared machine, is the example that Matt gave, where they don't have access to anything in your process except you are sharing a CPU, and so you're sharing the CPU's hierarchy of caches. So

apparently the attack involves them somehow getting the attacker able to get you to encrypt and decrypt specific information, and then them looking at modeling what's in the cache and using timing differences which they can theoretically detect in a shared environment in order to glean some information about the secret key which is loaded in there and doing your encryption/decryption.

So, okay. That's the second worst problem. And what it requires is that you have the volume mounted. You've already given your password. The system is up and running. And then somehow some malicious code has access to fine-grained analysis of the timing of your processor. So, okay. It would be better if, for example, the new Intel instructions were used, which are available probably moving forward to forks of TrueCrypt, because that allows them to speed up AES without using large tables. And so that makes AES more cache-timing neutral.

But remember that many purposes where you want your drive encrypted is the laptop turned off, sitting next to you at the airport, and you want to make sure no one can get into it. So understand that the nature of this attack is while it's in use, while the key is loaded. Remember that may be bad, if you've got malicious code, it's just looking at your hard drive. If it's able to read your drive, it's reading it after TrueCrypt has decrypted it. So again, this is an attack that you have to understand how high the bar is for exploitation, and that the typical use case is that you want your system when it's off to be absolutely safe from anyone accessing the drive. When it's on and running, presumably you're there and using it. Hopefully there's not bad stuff in there.

But, boy, even if you did have malware in your computer, it's easier for it just to read the drive because the drive is decrypted for it, just like it is for you. So that's the second worst problem. And again, in many cases, not to be - nothing that anyone needs to worry about, especially in the case of your secrets being kept when the system is not in use.

Now, these last two are, in fact, it's these guys who are paid to have problems even marked this one as "low severity" and "difficult." So not only is not bad, but it's hard to do. And that is, the keyfile mixing was found not to be cryptographically sound. And I've got to say, I don't understand this because this is just crazy.

Leo: If you don't understand it, we got no hope.

Steve: Well, what I mean is, I don't understand how the same people who engineered this other really good stuff could have done such a poor job with keyfile mixing. And my theory is that, where there was a model they could follow, like Peter Gutmann's entropy pool, or using the existing XTS mode of disk encryption, where there was a model to follow, they did. But unfortunately, when they invented something themselves, it was really lame.

So, okay. So what are keyfiles? Remember that we could either - okay. So the results of all of that cryptographic API, moving your mouse around like crazy until you're done, or tired, or the mouse wears out, that's all generating a key which needs to be very random because that's the master key which will be used to encrypt the drive's contents. Then your passphrase needs to be strong because it encrypts that key, that is, it encrypts the header. And it's the header on the volume that contains the encrypted master key, which all that entropy went in order to create. So inventing something else, they said, well, let's not just rely on the user's password. Let's let them use keyfiles. So this was a concept that may have been unique to TrueCrypt.

And TrueCrypt, remember, TrueCrypt had a predecessor. I'm trying to think. Was it DriveCrypt? There was something before TrueCrypt. I have it. I mean, I was using it way back then. And there was some argument over intellectual property rights and...

Leo: Oh, yeah, yeah, yeah.

Steve: ...people got upset with each other and wandered off in different directions. But so I don't know where keyfiles came from. But the idea was that, rather than just relying on what the user entered for their password, TrueCrypt could also be told, "Go look at this file over in this corner." We're not telling anybody, like we'll encrypt that in the header, too, so that, when I enter my password, the password decrypts the header. But then you know to go read a keyfile and get all of that gibberish and mix that in, in order to decrypt the master key for the volume. So the idea was it was just - you could use any other file or files in the system to provide more secrecy than just your own username or, I'm sorry, your own password.

So what they did was just lame. So in the auditors' description they said: "TrueCrypt allows the use of keyfiles that are included with the user's passphrase in the derivation of the key used to unlock a volume. However, TrueCrypt does not mix the keyfile content into the passphrase in a cryptographically sound manner." Which is just crazy. I mean, they just fabulously produced entropy pool because someone else had designed it. And unfortunately, left to their own devices - okay. So I'll give our listeners a sense. And those who are cryptographically astute are going to be groaning. A 64-byte buffer is constructed, initially zeroed. That's called the "keypool," that is used to hold the entropy generated from the keyfiles. For each keyfile, a maximum of 1K bytes - I'm sorry, 1024 kilobytes, one megabyte - are read.

A CRC, a cyclic redundancy check, initially all Fs, all ones, and using the polynomial [some random polynomial] 04c11db7, is constructed. So that's a polynomial for the CRC. And for each byte in the file, it is updated. Each time the CRC is updated, its four bytes are individually added into the keypool, modulo 256, and advancing, so the first time it updates bytes 0 through 3, the second time 3 through 7 - I wonder if they meant 3 through 7 or 4 through, anyway - and so on, wrapping around when it reaches 64. The keypool output at the end of the first keyfile is used as the input keypool for the second keyfile. Which just means they don't reinitialize that keypool if you give it multiple files.

So what I just read means that they just sort of made up a lame solution for turning a file into a 64-byte hash, essentially. But they didn't use a hash. They did this. And it isn't even time critical. So there's no reason. This doesn't need to be fast. And then it says after all the keyfiles are processed, each keypool byte is added, again modulo 256, into the user's password byte at that position. So added into the password byte. If the password is less than 64 bytes, the keypool byte in that position is used directly.

So then they wrote: "The use of CRC in this way is not cryptographically sound. When mixing entropy from multiple sources, an attacker who controls one source of entropy should not be able to fully negate or manipulate the other sources, even if the attacker is aware of what the other data is. The use of a cryptographic hash function is the correct way to mix entropy together. Assuming the hash function is unbroken, the best attack able to be mounted is a brute-force search for an input that, when combined with the uncontrolled input, yields a desirable output." Which is to say it's not possible, as far as we know.

So, okay. So that's my, I mean, the only way I can explain that something so randomly

awful as this was used is that they came up with this themselves. Or maybe it existed already. Maybe they inherited it from somewhere else, and they didn't care. Now, again, this doesn't matter because - okay. So what would this attack scenario be? These guys didn't even give one because there isn't one. It would have to be, while you were creating your volume and supplying keyfiles, some attacker was seeing that, reading the key files, running this algorithm, and then changed a keyfile on the fly to negate the effect of the keyfiles. And that's all I can imagine. I mean, it's like, okay. So basically these guys found something, you know, I mean...

Leo: It's bad in practice, but not...

Steve: Oh, it's beyond bad. They found a real turd in the middle of TrueCrypt, but in a place where it just absolutely doesn't matter. I mean, it just - no one cares. So it's like, okay. Again, it's embarrassing, and it's inexplicable because the rest of TrueCrypt seemed - maybe it was a summer intern who did this, and they never looked at the code. I don't know. And then, lastly, the fourth and least concerning is that they did not authenticate - again, this sort of substantiates my theory because they did not authenticate the volume headers.

So the volume header is the thing they invented which contains the secret key, the decryption key for the rest of the drive, and some other housekeeping information, the size of it and where it is and so forth. So the volume header and the volume data are encrypted separately because of course the drive is encrypted under the master key for the drive, and that's kept in the volume header, and it's the user's password that is used to encrypt the master key of the volume.

So again, they're using CRC32, and they're using, they've taken advantage of all this great cryptography throughout TrueCrypt, again, sort of where other people had solved the problems. But here they just use a - so they use a header format where they have a header with a magic string TRUE at the beginning of the volume header. Then they calculate a 32-bit CRC over the master key material. So they CRC check the master key. And then they do another CRC over the rest of the volume header.

And the problem is a CRC is not authentication. I mean, CRC is useful for checking an error that occurs in communications. The CRC checksum is what it really is. A cyclic redundancy check won't match. It's a high-accuracy, effective way of looking for, like, bits that were changed by noise. But any attacker could easily create a change that the CRC would think was fine. That is, you could easily defeat this. So they kind of, I guess, I mean, we could just say that it was unauthenticated because it's lame authentication, and the proper way to do this is, again, with just a message authentication code. We have all that crypto, all that technology. It's trivial to do. They didn't do it.

Now, does it matter? No, because what can an attacker do? I mean, it's the volume header. In there is the key, but it's encrypted. And so if they mess it up, then they just mess up the encryption. So again, nothing to worry about. What I've just given everyone are the big four problems. And as we can see, none of them are really a big concern.

So the takeaway is, you know, thank you for the audit. Everybody gained from this. The people who have forked TrueCrypt will certainly look at these things and fix them. And why not? I mean, they might as well be better. But in no way should anybody who is currently using TrueCrypt or would still want to use it in the future be dissuaded from doing so. None of these are deal killers.

Leo: And even more importantly, people going forward with forks have a really clear roadmap to make something that's a hundred percent perfect.

Steve: Yup, exactly.

Leo: That's well done.

Steve: Exactly, fix these problems, and then you can say we fixed the things the TrueCrypt audit found, and we've extended it in all kinds of other ways, too.

Leo: Of course then you have to have another audit of the new thing. Make sure they didn't introduce anything.

Steve: Yeah, well, yeah, in fact, where did these guys come from?

Leo: So here's the good news. If you go to - what was the website? Is TrueCrypt...

Steve: Auditedyet, is it, I think? Istruecryptauditedyet.

Leo: Let me just type it in and see, dotcom, and the answer is YES. That's nice. That's nice. And there's all the information, as well.

Steve: Yeah.

Leo: So it was a crowd-sourced project, Indiegogo, and the money was raised. And Matthew Green, is he part of this company that did it? Or what's the deal? It's a different...

Steve: No. No, no. So he's with Johns Hopkins.

Leo: Right.

Steve: And this is a group of three cryptographers, Tom Ritter among them. Tom has a site, I think it's Ritter on Cryptography [CiphersByRitter.com]. Tom is a well-known cryptographer. So it's just three guys who were hired and commissioned in order to perform this work.

Leo: So that's who iSEC Partners is.

Steve: Exactly.

Leo: Okay. All right.

Steve: That's those guys.

Leo: Got it.

Steve: The iSEC Partners, and they call themselves NCC or Crypto Group or something, so. And they produced that PDF containing all the details. So anyway, that's where we are. It's where I expected we would be. I'm glad for this. I think TrueCrypt, if it works, continues to be entirely useful. And in fact, I guess my feeling is I do wonder about the fork. That is, I'm staying with TrueCrypt.

Leo: Right. We know what the flaws are in TrueCrypt, and they're not significant.

Steve: And, yes, we don't know unless we do, unless we find out exactly every single change that has been made to TrueCrypt since then, you know, absolutely better not have introduced any more problems. And we know how easy it is to introduce problems. I mean, it's just - it's too easy.

Leo: Right, right.

Steve: It's difficult not to.

Leo: So in your opinion, none of these four flaws make a hill of beans' difference.

Steve: No, not at all.

Leo: Okay.

Steve: Exactly. And what'll happen is, I think, what is it, was it when Mac OS X 10.10 came out, TrueCrypt's volume or TrueCrypt's version parser saw it incorrectly and said, oh, I won't run on that. And I don't know about TrueCrypt on UEFI volumes. And certainly we're probably going to lose it when we switch to Secure Boot. So what's going to be happening is, its age is going to sort of age it out of use. But while it's useful, if it works for you, it's still my first choice.

Leo: Steve Gibson, you did it again. You made us feel better at the same time as scaring the pants off us. And I count on that in each and every episode. Now, that's because - I'm really glad. In fact, who would have thought we'd get here? I felt like

that was going to be a long - it doesn't solve the mystery at all.

Steve: No. We still - all we have is conjecture.

Leo: Yeah.

Steve: But as far as we know, one of the theories of their departure, which is they were afraid of the results of the audit, and so they wanted to distance themselves before that, that seems like absolutely zero likelihood now.

Leo: Steve is at GRC.com. That's where his SpinRite, the world's best hard drive, maintenance, and recovery utility lives. Go there, buy it, support Steve's fine work. But there's lots of free stuff there, including SQRL. We were talking the other day, I hate passwords. We've got to solve this. It's fascinating. He's got all sorts of stuff there.

Steve: Oh, I forgot to mention there was a nice TED Talk that was talking about authentication and identity and referred to SQRL as the solution to it. So...

Leo: That's great.

Steve: Yeah, so the word's getting out.

Leo: Is it on the TED site?

Steve: I thought I had it in my notes, but it must not have made the transfer over to my Google Doc. So I'll see if I can mention it next week.

Leo: Yeah. Or follow Steve on Twitter, @SGgrc. He has all the information there.

Steve: Yup.

Leo: You can also get to GRC.com for feedback. That's where we're going to take our questions for next week's feedback episode, GRC.com/feedback. He's got 16Kb versions of the audio there. He's got transcripts. Elaine's fine. She's writing - it was just an email glitch. She's writing away.

Steve: Oh, I forgot to say, yes, there was never any problem. I just didn't receive email from her. She sent it; it didn't get to me. And because I use receiving it to trigger all of the other downstream events, they didn't happen, either. So anyway, she just - she was surprised I hadn't received it. She sent it again, and then I immediately posted it.

Leo: Right on. Those transcripts are great. A lot of times it's easier to read along while you're listening to the show. We have higher quality video and audio of the show at our site, TWiT.tv/sn. We put it on YouTube, too, YouTube.com/securitynow. That's handy, I should point out, for anybody who wants to share a snippet. The easiest way to do it is to scrub in the YouTube video to the part you want to share with somebody, whether it's on Twitter or in an email or whatever. And then if you click the "share" button, it'll say, do you want to start at this point? You check that box, and the URL will have the minutes and seconds. And so you can actually share a little part of the show directly that way. I think a lot of people don't know that about YouTube, but that's one of the reason we put every show we do on YouTube so that you can do that, if you want to share a little bit of it.

You can also get it wherever you get your podcasts, iTunes, of course, and all those places, or your podcast app on your smartphone. And there are TWiT apps, too, thanks to our great and devoted community. We thank you. But that's about it. If you want to be back next week, we do this every Tuesday at 1:30 p.m. Pacific Daylight Time. That's 4:30 Eastern time, 2030 UTC, at live.twit.tv. Watching live is fun because, you know, you can chat in the background. We have a great bunch of people that are chatting. We also have some great people in the studio [crosstalk].

Steve: And we have our Q&A, we'll do a Q&A next week, as long as the world doesn't collapse in the meantime, in which case we'll cover the collapse. But GRC.com/feedback, and send your thoughts and questions and comments, and we'll share them and do a podcast about them.

Leo: Did you see John Oliver's interview with Edward Snowden?

Steve: You know, I didn't. I got a bunch of tweets about it.

Leo: It's good. It's on YouTube.

Steve: Okay, good, I'll check it out.

Leo: It's very funny. I think he cuts right to the core of the matter. Instead of, like, a lot of airy fairy conversation about what it means and everything, this is what he's so good at. He really brings it down to the personal level. And you'll know what I'm talking about.

Steve: Neat. I will absolutely watch it.

Leo: Yeah. Thanks, Steve. We'll see you next week.

Steve: Cool. Thanks, Leo.

Leo: Bye-bye.

Steve: Bye-bye.